

Optimal classification trees
D. Bertsimas and J. Dunn (2015)

Decision trees are often built in a (suboptimal) greedy way, unconstrained (to detect strong splits hidden behind weak ones) and pruned (to account for the complexity penalty).

The problem of finding the optimal tree of depth at most D can be formulated as mixed integer program:

- Binary variables for each node, indicating if there is a split;
- Binary variables to select the variable used for the split;
- Binary variables indicating non-empty leaves;
- Binary variables indicating, for each observation and each node, whether it goes to the left or to the right;
- Binary variables indicating the majority class in each leaf

(and many constraints).

For the hyperparameters:

- Progressively increase the tree depth, using CART or a shallower tree as a warm start;
- The complexity penalty can be formulated as a hard constraint on the number of nodes: progressively increase the number of nodes;
- Idem for the minimum number of observations in each leaf.

This can be generalized to multivariate decision trees (hyperplane splits, instead of axis-aligned ones), with greedy (single-split-optimal-tree based) warm start (or greedy logistic regressions).

**Learning optimal and fair decision trees
for non-discriminative decision-making**
S. Aghaei et al.

Another MIP formulation of the optimal decision (or regression) tree; also allowing fairness penalties.

Implementation in `ODTlearn`.

**Learning optimal classification trees
using a binary linear program formulation**
S. Verwer and Y. Zhang

Reformulation of the optimal decision tree MIP without constraints for each observation in the training set: just sum the constraints for the rows having the same feature values.

**Optimal constraint-based decision tree
induction from itemset lattices**
S. Nijssen and E. Fromont

DL8 uses frequent itemset mining to build optimal decision trees on binary data (every path, or a decision tree, can be mapped to an itemset): the best decision tree on a subset of the data is obtained by considering all possible ways of partitioning it in two, and recursively determining the best tree for each partition.

**Learning optimal decision trees
using caching branch-and-bound search**
G. Aglin et al.

Implementation improvements for DL8, in `PyDL8.5`

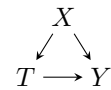
Optimal policy trees
M. Amram et al.

The optimal decision tree MIP can be generalized to build an optimal *policy* tree,

$$\text{Minimize}_{\tau: \mathcal{X} \rightarrow \{0,1\}} \sum_i \Gamma_{i,\tau(x_i)}$$

after estimating the counterfactuals with a doubly robust (DR) estimator

$$\Gamma_{it} = \frac{y_i - \hat{y}_{it}}{\hat{p}_{it}} \mathbf{1}_{z_i=t} + \hat{y}_{it}.$$



Optimal prescriptive trees
D. Bertsimas et al.

Joint (MIP) tree model for counterfactual estimation (locally constant or piecewise linear) and optimal treatment assignment.

Generalized random forests
S. Athey et al. (2016)

Generalized random forests fit a quantity $\theta(x)$ identified as the solution of local moment equations

$$\mathbb{E}[\psi_{\theta(x)}(X) | X = x] = 0.$$

The tree is built recursively, splitting a (parent) node P into (children) C_1, C_2 , maximizing

$$\Delta(C_1, C_2) = \frac{|C_1| \cdot |C_2|}{|P|^2} \left\| \hat{\theta}_{C_1} - \hat{\theta}_{C_2} \right\|^2,$$

which can be approximated with a Taylor expansion

$$\begin{aligned} A_P &= \text{Mean}_{i \in P} \nabla \psi_{\hat{\theta}_P}(x_i) \\ \rho_i &= -A_P^{-1} \psi_{\hat{\theta}_P}(x_i) \\ \tilde{\Delta}(C_1, C_2) &= \sum_{j \in \{1,2\}} \frac{1}{|C_j|} \left(\sum_{i \in C_j} \rho_i \right)^2. \end{aligned}$$

The final estimate is not obtained by averaging estimates from each tree, but by using the ensemble to compute similarity weights $\alpha_i(x)$ between the new sample x and training samples i and putting them in the empirical estimating equation

$$\hat{\theta}(x) = \underset{\theta}{\text{Argmin}} \left\| \sum_i \alpha_i(x) \psi_{\theta}(x_i) \right\|_2.$$

R implementation in `grf`.

evtree: evolutionary learning of globally optimal classification and regression trees in R
T. Grubinger et al. (2014)

Decision trees are usually built greedily: they are only locally optimal – genetic algorithms can search for a globally optimal tree.

Parties, models, mobsters: a new implementation of model-based recursive partitioning in R
A. Zeileis and T. Hothorn

Model-based recursive partitioning (MOB) is a “decision” tree with models (e.g., linear models) in the leaves. They can use different variables for the models $y \sim x_1 + \dots + x_m$ and the splits z_1, \dots, z_ℓ . The tree is built recursively:

- Fit the model to the current node;
- Test for parameter instability (SupLM fluctuation test) to select the variable to split on;
- Use exhaustive search to select the split point;
- Split the node, and iterate until the desired depth;
- Prune the tree by looking at the AIC or BIC improvement.

R implementation in `partykit`.

Generalized M-fluctuation tests for parameter instability
A. Zeileis and K. Hornik (2007)

Given observations $Y_i \stackrel{\text{indep}}{\sim} F_{\theta_i}$, we can test

$$H_0 : \forall i \theta_i = \theta_0$$

$$H_1 : \theta_i \text{ varies over time}$$

using a score function ψ , *i.e.*, a function such that

$$\mathbb{E}_{Y \sim F_\theta} [\psi_\theta(Y)] = 0,$$

e.g.,

$$\psi_\theta(y) = \frac{\partial \text{loss}(y; \theta)}{\partial \theta}$$

where the loss is the log-likelihood or the residual sum of squares.

Since $\hat{B}_n^{-1/2} W_n(\cdot, \hat{\theta}_n) \xrightarrow{d} W^0(\cdot)$, where

- n is the number of observations;
- $\hat{\theta}_n$ is estimated using the whole sample:

$$\sum_{i=1}^n \psi_{\hat{\theta}_n}(Y_i) = 0;$$

$$\cdot W_n(t, \theta) = \frac{1}{\sqrt{n}} \sum_{i=1}^{\lfloor nt \rfloor} \psi_\theta(Y_i);$$

$$\cdot B(\theta) = \text{Var}_{Y \sim F_\theta} [\psi_\theta(Y)];$$

$$\cdot \hat{B}_n = \frac{1}{n} \sum_{i=1}^n \psi_{\hat{\theta}_n}(Y_i) \psi_{\hat{\theta}_n}(Y_i)^\top;$$

$$\cdot W^0(t) = W(t) - tW(1) \text{ Brownian bridge};$$

$$\cdot W \text{ is a standard Brownian motion,}$$

we can use

$$\text{SupLM} = \sup_{t \in \Pi} \frac{\|\text{efp}(t)\|_2^2}{t(1-t)}$$

$\text{efp}(t) = \hat{B}_n^{-1/2} W_n(t, \hat{\theta}_n)$ (empirical fluctuation process)

$\Pi \subset [0, 1]$ interval with suspected break

as test statistic (and approximate p -values). For linear regression,

$$\psi_B(y_i, x_i) = x_i(y_i - x_i^\top \beta).$$

Approximate asymptotic p-values for structural change tests
B.E. Hansen (1997)

P -values for the limiting distribution

$$\sup_{t \in \Pi} \frac{\|W^0(t)\|_2^2}{t(1-t)}.$$

ctree: conditional inference trees
T. Hothorn et al.

Decision tree using statistical tests (adjusted for multiple testing) to select the splits (instead of heuristic, overfitting-prone criteria).

Fast optimal leaf ordering for hierarchical clustering
Z. Bar-Joseph et al. (2001)

Approximation trees: statistical stability in model distillation
Y. Zhou et al.

To explain a blackbox model (e.g., a random forest), distill it into a decision tree using pseudo observations (real observations plus Gaussian noise, for a user-chosen bandwidth – *i.e.*, samples from a KDE of the data). Choose the number of pseudo observations so that the trees be stable – use a statistical test on the Gini index (which is used to select the splits). Choose the tree depth by comparing the variance of the observations in a leaf with the variance of the blackbox model.

Constructing optimal L_∞ star discrepancy sets
F. Clément et al. (2024)

The L^∞ star discrepancy of a finite set $P \subset [0, 1]^d$ is

$$d_\infty^*(P) = \sup_{q \in [0, 1]} \left| \frac{|P \cap [0, q]|}{|P|} - \lambda([0, q]) \right|$$

where λ is the Lebesgue measure. It suffices to consider $q \in \bar{\Gamma}(P)$.

$$\bar{\Gamma}(P) = \bar{\Gamma}_1(P) \times \dots \times \bar{\Gamma}_d(P)$$

$$\bar{\Gamma}_i(P) = \Gamma_i(P) \cup \{1\}$$

$$\Gamma_i(P) = \{x_i : x \in P\}$$

Strong laws for L and U statistics
J. Aaronson et al. (1996)

U-statistics are averages of kernels of random samples, *i.e.*, they are of the form

$$U_n(X_1, \dots, X_n) = \text{Mean}_{\substack{I \subset [1, n] \\ |I|=r}} f(X_I)$$

where f is a symmetric function of r variables, *e.g.*, $f(x_1) = x_1$ for the mean, or $f(x_1, x_2) = \frac{1}{2}(x_1 - x_2)^2$ for the variance.

L-statistics are linear combinations of order statistics, *i.e.*, they are of the form

$$L_n(X_1, \dots, X_n) = \sum_{i=1}^n \mu \left(\left[\frac{i-1}{n}, \frac{i}{n} \right] \right) X_{n;i},$$

e.g.,

density	statistic $L(F) = \int_0^1 F^{-1} d\mu$
$J(u)$	$M_1(F) = E[X]$
$J(u) = \alpha u^{\alpha-1}$	$P_\alpha(F) = \int x dF^\alpha(x)$
$J(u) = 4u - 2$	$g_1(F) = E X - X' $.

Relevance-based importance: a comprehensive measure of variable importance in prediction
M. Czaronis et al. (2024)

The adjusted fit, from a relevance grid model, defines function

$$\begin{aligned} \mathcal{P}(\text{predictors}) \& \longrightarrow \mathbf{R} \\ \theta & \longmapsto \text{Fit}_{\theta,t}^{\text{adj}} \end{aligned}$$

for each observation t ; the *relevance-based importance* $\text{RBI}_{t,k}$ is the corresponding Shapley value for predictor k .

Learning optimal Bayesian networks: a shortest path perspective
C. Yuan and B. Malone (2013)

One can find the causal graph minimizing

$$\text{Score}(G) = \sum_i \text{Score}(x_i, \text{Pa}_i)$$

with dynamic programming, adding successor nodes (leaves) one by one (finding a path from \emptyset to V in the "order graph" – the Hasse diagram of the nodes)

$$\text{Score}(V) = \min_{X \in V} [\text{Score}(X \setminus \{X\}) + \text{BestScore}(X, V \setminus \{X\})]$$

$$\text{BestScore}(X, V) = \min_{P \subset V} \text{Score}(X, P)$$

or with A^* , with a heuristic relaxing the acyclicity constrain

$$h(U) = \sum_{X \in V \setminus U} \text{BestScore}(X, V \setminus \{X\})$$

(better heuristics exist).

$$\begin{array}{c} V \\ | \\ \text{BestScore}(X, V) \\ | \\ V \cup \{X\} \end{array}$$

Learning optimal Bayesian networks using A^* search
C. Yuan et al. (2011)

FastText.zip: compressing text classification models
A. Joulin et al.

Linear models for text classification with n -gram features are easy to train, perform well, but are very large:

- Use product quantization (PQ: k -means on subsets of the coordinates) or orthogonal product quantization (OPQ: PQ after applying a learned rotation), separating norm and direction of the vectors (the norms span 3 orders of magnitude), and retrain the layers after quantization;
- Prune the vocabulary (largest weightes, ensuring there is at least an n -gram in each document); keep the n -grams selected in a **set** (a Bloom filter degrades performance);
- Use Vowpal Wabbit's hashing trick to store the weights.

Enriching word vectors with subword information
P. Bojanowski et al.

Represent words as bags of character n -grams ($3 \leq n \leq 6$); use skipgram to learn the subword representations and add them.

Bag of tricks for efficient text classification
A. Joulin et al.

fastText is a linear model for text classification, using

- Word and n -gram embeddings as features;
- A low-rank constraint

$$\text{Maximize}_{A,B} \text{Mean}_n y_n \log \text{softmax}(BAx_n)$$

- If there are many classes, a hierarchical softmax based on the Huffman coding tree.

Optimized product quantization

T. Ge et al.

Product quantization (k -means on M subsets of the coordinates, for approximate nearest neighbour (ANN) search) after a learned orthogonal transformation:

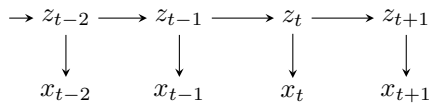
- Alternatively estimate the cluster centers (k -means) and the orthogonal matrix (Procrustes);
- Initialize with the following heuristic (approximation of the exact solution for Gaussian data):
 - Compute the PCA and sort the eigenvalues in decreasing order;
 - Start with M empty buckets;
 - Sequentially pick the largest eigenvalue and assign it to the bucket with the minimum product of eigenvalues (unless it is already full);

this gives k^M codewords.

Diffusion forcing: next token prediction meets full sequence diffusion

B. Chen et al.

Combine Bayesian updates



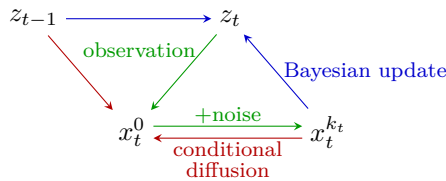
with diffusion

$$x^0 \xrightleftharpoons[\text{denoising}]{+\text{noise}} x^k$$

or conditional diffusion

trying to forecast the added noise $\varepsilon_t = x_t^{k_t} - x_t^0$ from z_{t-1} , $x_t^{k_t}$ and k_t .

One can sample on a grid, time \times noise, with an arbitrary (time-dependent) noise schedule – noise can be seen as a form of masking.



Classifier-free diffusion guidance

J. Ho and T. Salimans (2021)

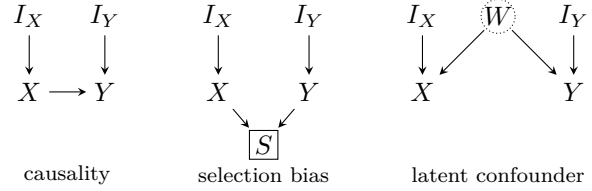
Classifier guidance is a diffusion model whose score is combined with the gradient of a classifier – but we cannot use an off-the-shelf classifier: it has to be trained on noisy data (this is reminiscent of GANs). Instead, jointly train a conditional and an unconditional diffusion model, and combine their scores

$$(1 + w)\varepsilon_\theta(z, c) - w\varepsilon_\theta(z).$$

Gene regulatory network inference in the presence of selection bias and latent confounders

G. Luo et al.

With interventional data ($I_X \rightarrow X$ and/or $I_Y \rightarrow Y$), it is possible to distinguish between



with conditional independence tests.

DeepSeek-R1: incentivizing reasoning capability in LLMs via reinforcement learning

DeepSeek-R1-Zero uses RL (GRPO) on DeepSeek-V3-Base to enhance its reasoning capabilities:

- Prompts: unspecified (math, leetcode, etc.);
- Reward: only accuracy (correct final answer) and format, as specified by the prompt: `<think>...</think>` and `<answer>...</answer>`.

During training, the model spends more and more time thinking, but the `<think>` section becomes less and less readable (it mixes Chinese and English).

DeepSeek-R1 uses 4 steps:

- SUPervised fine-tuning (SFT) with high-quality data, e.g., readable answers from DeepSeek-R1-Zero;
- RL, as above;
- SFT, with 600k reasoning samples (ask the model to generate several answers, and have DeepSeek-V3 select the best) and 200k non-reasoning samples from (DeepSeek-V3, unspecified prompts);
- RL, for alignment (unspecified prompts and rewards).

DeepSeekMath: pushing the limits of mathematical reasoning in open language models

Z. Shao et al.

Start with a code model, and fine-tune it on a large mathematical corpus:

- Train a FastText classifier on OpenWebMath;
- Apply it to CommonCrawl;
- Use the results to identify websites with math contents;
- Add the corresponding pages to the FastText training data (adding arxiv papers does not bring any improvement); re-train;
- To avoid benchmark contamination, exclude pages containing a 10-gram from one of the benchmarks;
- Apply the new model to CommonCrawl.

Also evaluate on miniF2F (convert an informal proof into a formal one, checked with Isabelle) and on GSM8K (math with tool use: Python, SymPy).

In PPO,

$$J = \mathbb{E}_{\substack{q \sim \text{queries} \\ o \sim \pi_{\text{old}}(\cdot|q)}} \text{Mean}_{1 \leq t \leq |o|} \frac{\pi_{\theta}(o_t|q, o_{<t})}{\pi_{\text{old}}(o_t|q, o_{<t})} - \beta \text{KL}(\pi_{\theta} \parallel \pi_{\text{ref}})$$

GRPO (group relative polici optimization) replaces the advantage A with the average reward from π_{old} ,

$$A_i = \frac{r_i - \text{mean}(\mathbf{r})}{\text{std}(\mathbf{r})}$$

and uses

$$\text{KL}(\pi_{\theta} \parallel \pi_{\text{ref}}) = \frac{\pi_{\text{ref}}(\cdots)}{\pi_{\theta}(\cdots)} - \log \frac{\pi_{\text{ref}}(\cdots)}{\pi_{\theta}(\cdots)} - 1.$$

DeepSeek-V3 technical report

DeepSeek-V3 combines:

- MoE (top-k and shared experts);
- No auxilliary loss to ensure load balancing between experts replaced by a bias term for each expert;
- Sequence-wise auxilliary loss;
- Multi-head laent attentio (MLA): joint low-rank compression of K and V to reduce the size of the KV cache;
- Multi-token predictin;
- FP8 mixed precision (FP8, BF16, FP32);
- Context length extension (first 32k, then 128k);
- Many parallelism and communication improvements).

Training required 2 months on 2048 GPUs (\$5m).

Softmax is not enough (for sharp out-of-distribution)

P. Veličković et al.

“Softmax” should have been called “softargmax”.

A function is *sharp* if it only depends on a constant number of inputs, e.g., max. Softmax cannot approximate shaprness with increasing problem size: use a temperature-adjusted softmax instead, with the temperature depending on the entropy of the initial probability distribution.

Data-driven discovery of dynamical systems in pharmacology using large language models

S. Holt et al.

LLM-based, CMA-ES-like optimization to find the ODE (Python code) best fitting the data; the LLM can also request more features (from a list). Iterate:

$$\begin{aligned} f &= \text{LLM}(\text{descriptions, past attempts}) \\ \ell &= \text{Min}_{\theta} \text{loss}[f(\text{Data}; \theta)]. \end{aligned}$$

Rethinking early stopping: refine, then calibrate **E. Berta et al.**

(Multiclass) classifiers minimize two quantities:

- Refinement error (whether the classes are in the correct order);
- Calibration error (whether the probabilities are correct).

They are not simultaneously minimized during training. Stop training when the refinement loss stops decreasing,

$$\text{Min}_g \text{Mean}_{(X,Y) \sim \text{Data}} \text{loss}[g(f(X), Y)]$$

where g is the relibration – temperature scaling is good enough.

Towards safe reinforcement learning via constraining conditional value at risk **C. Ying et al.**

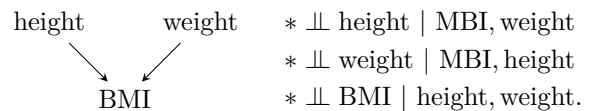
CVaR-PPO (aka CPPO) adds a CVaR constraint to the PPO loss, transforms it using the formulation of CVaR as an optimization problem, and replaces the hard constraint with its Lagrangian relaxation.

FinRL-DeepSeek: LLM-infused risk-sensitive reinforcement learning for trading agents **M. Benhenda**

RL trading agent, using technical indicators and LLM-derived signals (trade recommendation from news, confidence in this recommendation, risk assessment, each as a numeric score in $[0, 10]$), trained with CVaR-PPO.

On causal discovery in presence of deterministic relations **L. Li et al.**

Deterministic relations create independence and wreak havoc in constraint-based causal discovery algorithms: for instance, if $X \rightarrow Y$ is deterministic, then y is constant when conditioned on X , and therefore independent from everything: $* \perp\!\!\!\perp Y \mid X$. Similarly,



GES can be modified to account for deterministic relations:

- Identify deterministic variables (by looking at the variance of the residuals of regressions) and ignore independence relations coming from determinism;
- Use a (RKHS) ridge-adjusted BIC as score (because the variance matrix is singular in presence of determinism).

**Generalized score functions
for causal discovery
B. Huang et al. (2018)**

Using linear models in GES can introduce spurious relations, e.g., in

$$\begin{array}{ll} X_1 = \varepsilon_1 & X_1 \\ X_2 = X_1 + X_1^2 + \varepsilon_2 & \downarrow \\ X_3 = X_2 + X_2^2 + \varepsilon_3 & X_2 \\ & \downarrow \\ & X_3 \end{array}$$

the influence of X_1 on X_3 cannot be blocked by a linear function of X_2 , and can miss relations, e.g.,

$$\begin{array}{ll} X_1 = \varepsilon_1 & X_1 \\ X_2 = (\sin X_1 + \varepsilon_2)^2 & \downarrow \\ & X_2 \end{array}$$

Use a kernel conditional independence test to rule out potential parents of a node.

For the score function, use the (ridge-regularized) CV-log-likelihood or marginal likelihood of an RKHS regression (regression on

$$\begin{bmatrix} \kappa(x_1, x) \\ \vdots \\ \kappa(x_n, x) \end{bmatrix} \in \mathbf{R}^n$$

instead of $\phi(x) \in \mathcal{H}$).

**Universality, characteristic kernels
and RKHS embeddings of measures
B.K. Sriperumbudur et al.**

A kernel on X is a positive definite function $\kappa : X \times X \rightarrow \mathbf{R}$. I defines a subspace

$$\text{Span}\{\kappa(\cdot, x), x \in X\} \subset \mathcal{F}(X, \mathbf{R}).$$

It is *universal* if

$$\mathcal{H}_0 = \overline{\text{Span}\{\kappa(\cdot, x), x \in X_0\}} = \mathcal{F},$$

for some $X_0 \subset X$ and some Hilbert space $(\mathcal{F}, \|\cdot\|)$ of functions on X .

It is *characteristic* if

$$\begin{cases} \text{Borel}(X) & \longrightarrow \mathcal{H} \\ P & \longmapsto \int_X \kappa(\cdot, x) dP(x) \end{cases}$$

is injective.

Examples include:

- $\mathcal{C}^0(X)$ and $\|\cdot\|_\infty$;
- $\mathcal{C}^0(X)$ and compact convergence;
- $\mathcal{C}_0(X)$ (continuous functions vanishing at infinity) and $\|\cdot\|_\infty$;
- L^0 and $\|\cdot\|_p$

**DoWhy-GCM: an extension of DoWhy
for causal inference in graphical causal models
P. Blöbaum et al.**

dowhy was limited to effect estimation

$$\mathbb{E}[Y \mid \text{do}(T = 1)];$$

with **dowhy-gcm**, it can answer causal queries: counterfactuals, direct and indirect effects, causal strength, root cause analysis, etc.

**Can large language models
infer causation from correlation?
Z. Jin et al.**

Can an LLM learn the PC algorithm, *i.e.*, infer causation from correlations (independence and conditional independence relations, expressed as text)? No. Fine-tuning does not help much.

**Causal inference using LLM-guided discovery
A. Vashishtha et al.**

Do not ask the LLM to find the whole causal graph (it would struggle to distinguish direct from indirect relations) but only the partial order it induces (the *transitive reduction*, aka Hasse diagram it induces).

You can compare DAGs with the L^1 distance between their transitive reductions.

Use a triplet prompt to infer this order: provide 3 variables and ask for their causal graph.

Applications include

- Post-processing the output of the PC algorithm, to orient the remaining unoriented edges;
- Constraints (transitive closure) for the PC or GES algorithms.

**Causal order: the key to leveraging
imperfect experts in causal inference
A. Vashishtha et al. (2024)**

Building a causal graph by asking pairwise questions to an LLM creates spurious edges – with pairwise queries alone, the LLM cannot distinguish between direct and indirect effects.

The not build a DAG, but just a *partial order*: it can be used

- To obtain a valid adjustment set (the ancestors of a node in a topological order);
- As constraints or prior for classical causal discovery algorithms.

Do not use pairwise queries, but triplet queries (either all triplets, or $k = 10$ triplets for each pair if the graph is large), aggregated with a majority vote: this creates fewer cycles.

The *topological divergence* between a partial order \preceq and a DAG defined by its adjacency matrix is the num-

ber of edges incompatible with the order,

$$D(\preceq, A) = \sum_{i \not\preceq j} A_{ij}.$$

***From causal to concept-based
representation learning***

G. Rajendran et al.

With no additional assumptions or data, we cannot recover latent factors Z from observed data $X = f(Z)$. If Z is a causal representation, and if we have interventions on all latent variables Z_j , it is identifiable – but that is a lot of interventions. A *concept* is a projection AZ of the latent factors; concepts are identifiable through conditioning.

***CLadder: assessing causal reasoning
in language models***

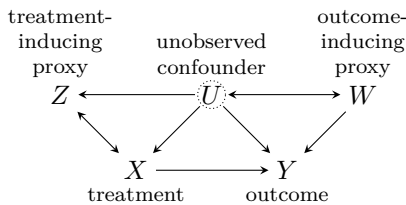
Z. Jin et al.

Dataset to check if LLMs already know do-calculus and can answer queries on the three rungs of the causality ladder (association, intervention, counterfactuals), without resorting to commonsense.

***Deep proxy causal learning and its application
to confounded bandit policy evaluation***

L. Xu et al. (2024)

PCL estimates causal effects in presence of unobserved confounders using proxies.



$$\text{ATE}(a) = \mathbb{E}_U[\mathbb{E}[Y | A = a, U]]$$

$$h \text{ s.t. } \mathbb{E}[Y | A = a, Z = z] = \int h(a, w) \rho_W(w | A = a, Z = z) dw$$

$$\text{ATE}(a) = \mathbb{E}_W[h(a, W)]$$

The “bridge function” h can be found with 2-stage regression.

***Causal reasoning and large language models:
opening a new frontier for causality***

E. Kıcıman et al.

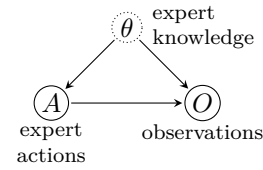
LLMs are good, but have unpredictable failures.

***Shaking the foundations: delusions in
sequence models for interaction and control***

P.A. Ortega et al.

In imitation learning, the model takes its own actions as evidence about the world: this leads to self-delusion.

Actions are causal interventions: do not model $P[O|A]$, but $P[O|\text{do}(A)]$.



***On information transfer
in control dynamical systems***

S. Sinha and U. Vaidya (2018)

The directed information is

$$I(X \rightarrow Y) = H(Y_1, \dots, Y_T) - \sum_{1 \leq t \leq T} H(Y_t | Y_{<t}, X_{\leq t}).$$

The transfer entropy is

$$T_{X \rightarrow Y} = H(Y_t | Y_{<t}) - H(Y_t | Y_{<t}, X_{<t}).$$

A dynamical system

$$x_{t+1} = f(x_t, y_t) + \text{noise}$$

$$y_{t+1} = g(x_t, y_t) + \text{noise}$$

defines a conditional probability distribution $\rho(y_{t+1}|y_t)$. Similarly, if we freeze x ,

$$x_{t+1} = x_t$$

$$y_{t+1} = g(x_t, y_t) + \text{noise}$$

defines $\rho_{\mathcal{F}}(y_{t+1}|y_t)$. The information transfer is

$$T_{x \rightarrow y}_t^{t+1} = H(\rho(y_{t+1}|y_t)) - H(\rho_{\mathcal{F}}(y_{t+1}|y_t)).$$

***Context is key: a benchmark for forecasting
with essential textual information***

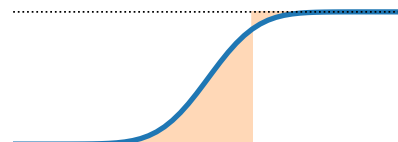
A.R. Williams et al.

Cik is a benchmark for *multimodal* probabilistic time series forecasting models (models also provided with a textual context for each time series), such as UniTime, Time-LLM, LagLlama, Chronos, Moira, TimeGEN.

The CRPS (continuous ranked probability score)

$$\text{CRPS}(F, x) = \int_{-\infty}^{+\infty} (F(y) - \mathbf{1}_{y \geq x})^2 dy \quad (\text{dim} = 1)$$

$$= \mathbb{E}_{X \sim F} |X - x| - \frac{1}{2} \mathbb{E}_{X, Y \sim F} |X - Y|$$



(the energy score uses $\|\cdot\|_2^2$ instead of $|\cdot|$) is a proper scoring rule generalizing the MAE to assess *probabilistic* forecasts – cross-entropy

$$- \text{Mean} \log f(x) = H(\text{data}, F)$$

$$H(p, q) = - \mathbb{E}_{X \sim p} \log q(x)$$

is often a better choice if the forecasted density is available.

The RCRPS (region-of-interest CRPS) is a weighted and penalized CRPS: to give more weight to some forecasting horizons and penalize (problem-specific) constraint violations.

Regions of reliability in the evaluation of multivariate probabilistic forecasts
É. Marcotte et al. (2023)

A *scoring rule* is a function S measuring how unlikely an event $y \sim \text{Data}$ is, wrt a forecasted distribution F ; it is *proper* if

$$\mathbb{E}_{y \sim \text{Data}} S(y, F)$$

is minimal for $F = \text{Data}$.

In practice, however, we do not have a ground truth distribution, but only a finite sample from it: for some combinations of

d : dimension

n : ground truth sample size

m : forecast sample size

the statistical power of most scoring rules to detect

- Differences in mean;
- Positive correlations;
- Skewness;
- Difference in marginal variance;
- etc.

is low – you need to increase n and m .

Time-MoE: billion-scale time series foundation models with mixture of experts
X. Shi et al.

Transformer (with SwiGLU, RMSNorm, no biases, ROPE, multi-resolution forecasting, MoE), trained on 300B data points.

Towards transparent time series forecasting
K. Kacprzyk et al.

TO make time series forecasting (from static features) “transparent”, let the user interactively see the impact of changing some of the inputs on:

- The shape of the forecast (a sequence of segments, labeled increasing/constant/decreasing and convex/linear/concave);
- Some properties of the forecast (mean, slope, maximum, minimum).

Fit a model with clubic splines (piecewise polynomials) $\mathbf{R}^k \rightarrow \mathbf{R}^T$.

Forecast evaluation for data scientists: common pitfalls and best practices
H. Hewamalage et al.

Long list of error measures for time series forecasting, with their suggested use and pitfalls.

- Use a simple baseline (naive, seasonal naive);
- Forecast plots $t, \hat{y} \sim \text{time}$ are misleading;
- Beware of non-stationarity (seasonality, trend, unit root (stochastic trend), heteroskedasticity, structural breaks) and non-normality (symmetry, tails, outliers, intermittency (zero-inflation));
- Beware of ddata leakage.

Chronos: learning the language of time series
A.F. Ansari et al.

T5-based foundation time series forecasting model, processing time series that have been mean-scaled (divide by the average absolute value) and quantized (uniform binning), trained with data augmentation: TSMixup, KernelSynth (Gaussian processes with kernels from the automated statistician).

The effectiveness of discretization in forecasting: an empirical study on neural time series models
S. Rabanset et al.

Binning input and output data tends to improve the performance of deep learning time series forecasting models (MLP, CNN, RNN).

Maximum entropy bootstrap for time series: the meboot R package
H.D. Vinod and J. López-de-Lacalle

The *maximum entropy bootstrap*, for potentially non-stationary time series,

- Separates the order of the observations from their distribution;
- Resamples the observations, using a piecewise uniform distribution (more precisely, a piecewise maximum entropy distribution, with a pre-specified mean on each interval, to allow values slightly more extreme than observed);
- Keep the order (sic).

Maximum entropy ensembles for time series inference in economics
H.D. Vinod (2006)

The uncertainty of machine learning predictions in asset pricing
Y. Liao et al. (2025)

There are many types of bootstraps:

- Classical: resampling with replacement;
- Residual: fit a model and esample the residuals;
- Wild: idem, after multiplying the residuals by $N(0, 1)$ (for heteroskedastic data);
- Blocks: blocks of constant sizes;
- Stationary: blocks of random length, from a geometric (exponential) distribution;
- Parametric: $X \sim F_{\hat{\theta}}$;
- Smooth: $X \sim \text{kde}$;
- etc.

To compute prediction confidence intervals for deep learning models on panel data, use a variant of the wild bootstrap:

$$\begin{aligned}\hat{y}_{it} &= \hat{g}(x_{i,t-1}) && \text{forecast} \\ \varepsilon_{it} &= y_{it} - \hat{y}_{it} && \text{residual} \\ \eta_t &\sim N(0, 1) && \eta_t, \text{ not } \eta_{it} \text{ or } \eta_i \\ y_{it}^* &= \hat{y}_{it} + \eta_t \varepsilon_{it}\end{aligned}$$

and fine-tune the model (starting from the trained one) for k epochs.

Applications include worst-case (uncertainty-aware) portfolio optimization

$$\text{Maximize}_w \text{Min}_{\mu \in \text{CI}} w' \mu - \frac{\lambda}{2} w' V w.$$

The stationary bootstrap **D.N. Politis and J.P. Romano**

Resample blocks of random lengths, from a geometric distribution.

Proceedings of the 7th annual AI in finance conference **Wolfe, 2025**

LLM applications in finance include:

- Comparison of ChatGPT-generated answers with human ones in conference calls – differences correlate with abnormal volumes;
- Denoising reported earnings;
- Human-free surveys, with synthetic personas (mimicking human participants);
- Making discretionary investing systematic;
- Identifying (and betting against) retain investor biases (technical analysis, from StockTwits).

Regimes **A. Mulliner et al. (2025)**

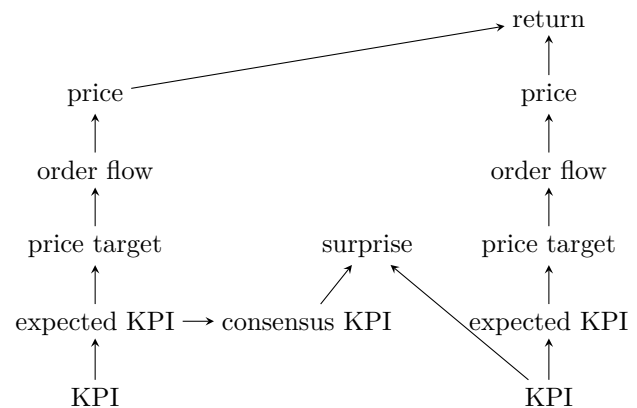
- Take a few macro-economic variables (S&P 500, 10y-3m, 3m, oil, copper, VIX, stock-bond correlation);
- Compute the (Euclidean) distance between today's state and past states;
- Buy (sell) assets with positive (negative) average returns in similar past situations.

One can also look at the most dissimilar past states, and take the opposite actions. [It works better with the Mahalanobis distance; a learned Mahalanobis distance gives similar results out-of-sample but overfits in-sample.]

Dynamic asset allocation using machine learning: seeing the forest for the trees **C. Mueller-Glissmann and A. Ferrario**

A *student tree* is a complex model (e.g., a random forest) distilled into a tree (on real data + noise).

Working backwards from returns **A. Kelleher and A. Mittal**



Mathematical models in epidemiology **F. Brauer et al. (2019)**

Automatic posterior transformation for likelihood-free inference **D.S. Greenberg et al. (2019)**

The Bayesian approach to statistics estimates the posterior $p(\theta|x)$ from the prior $p(\theta)$ and the likelihood $p(x|\theta)$.

Approximate Bayesian computation (ABC, aka likelihood-free inference) only assumes that we know the prior $p(\theta)$ and that we can sample from $p(x|\theta)$ – but we do not know the likelihood.

One can model the posterior as $q_{F(x,\phi)}(\theta) \approx p(\theta|x)$, where q_ψ is a family of tractable distributions and F is a neural net, trained with generated pairs (θ, x) ,

$$\text{Maximize}_{\phi} \sum_i \log q_{F(x_i, \phi)}(\theta_i).$$

Since we are interested in the posterior $p(\theta|x_0)$ at a specific point x_0 , we may want to restrict the θ samples to “informative” values (sampling θ from $q_{F(x_0, \phi_{n-1})}(\theta)$ instead of $p(\theta)$ to estimate ϕ_n) – this gives a biased posterior, but it can be fixed.

Deep symbolic regression: recovering mathematical expressions from data via risk-seeking policy gradients **B.K. Petersen et al.**

Symbolic regression usually uses genetic programming. Instead, use RL to train an RNN model, to generate a symbolic expression tree in pre-order traversal (depth-first, left-to-right). The RNN is not only fed the previous token, but also the parent and sibling nodes. The reward is the normalized root mean square error. The constant token is replaced with the constant maximizing the reward (using BFGS). The REINFORCE policy gradient (with baseline, for variance reduction) would maximize the expected sum of rewards – instead, we

want to maximize the maximum reward: this can be achieved by maximizing the conditional expected reward

$$\mathbb{E}_{\tau \sim p_\theta} [R(\tau) \mid R_\tau \geq R_\varepsilon(\theta)].$$

Why do tree-based models still outperform deep learning on tabular data?
L. Grinsztajn et al.

Neural networks:

- Are biased towards overly smooth functions;
- Are more affected (than tree-based methods) by uninformative features;
- Are rotation-invariant (rotating the input does not affect training).

KwikBucks: correlation clustering with cheap-weak and expensive-strong signals
S. Silwal et al. (2023)

Given a graph (V, E) , *correlation clustering* finds a clustering C minimizing

$$\sum_{(i,j) \notin E} C_{ij} + \sum_{(i,j) \in E} (1 - C_{ij}).$$

KwikCluster picks a vertex at random, forms a cluster by adding its neighbours, removes it, and iterates. Alternatively, find a *maximum independent set* and assign (in parallel) the other nodes to their first neighbour in that set – these are 3-approximations.

This can be generalized to budgeted correlation clustering – we have a limited number of queries to the exact graph, and an unlimited number to an approximation of the graph.

Spectral sparsification of graphs: theory and algorithms
J. Baston et al. (2009)

The *Laplacian quadratic form* of a weighted graph (V, w) is

$$Q_G(x) = \sum_{(u,v) \in E} w_{uv} (x_u - x_v)^2.$$

The *cut* of a subset of nodes $S \subset V$ is

$$\text{Cut}_G(S) = \sum_{\substack{u \in S \\ v \notin S}} w_{uv} = Q(\mathbf{1}_S).$$

Two weighted graphs are *cut-similar*, resp. *spectrally similar* if

$$\begin{aligned} \forall S \quad \sigma^{-1} \cdot \text{Cut}_{\tilde{G}}(S) &\leq \text{Cut}_G(S) \leq \sigma \cdot \text{Cut}_{\tilde{G}}(S) \\ \forall S \quad \sigma^{-1} \cdot Q_{\tilde{G}}(S) &\leq Q_G(S) \leq \sigma \cdot Q_{\tilde{G}}(S). \end{aligned}$$

The *effective resistance* between two nodes u, v is

$$R_{uv} = \left(\min_{x: x_u=1, x_v=0} Q_G(x) \right)^{-1}$$

(the equivalent resistance if you replace each edge with a resistor of resistance $1/w$).

The hypercube is a spectral approximation (a spectral sparsifier) of the complete graph on 2^n vertices (set the weights on the hypercube to $n/2\sqrt{\log n}$).

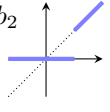
One can obtain spectral sparsifiers:

- By sampling q edges (with replacement) with probabilities $p_e \propto w_e R_e$ and assigning them weights $w_e/q p_e$;
- Or as a union of random spanning trees.

Spectral sparsification of graphs
D.A. Spielman and S.H. Teng (2010)

Gemma Scope: open sparse autoencoders everywhere all at once on Gemma 2
T. Lieberum et al.

To explain an LLM, train a *sparse autoencoder* (SAE, sparse and nonnegative) on its activations (after the MLP, or the activations on the residual stream – for all layers)

$$\begin{aligned} x \mapsto y &= \sigma(W_1 x + b_1) \mapsto \hat{x} = W_2 y + b_2 \\ \sigma(z) &= \text{JumpReLU}_\theta(z) = z \odot \mathbf{1}_{z \geq \theta} = \begin{array}{c} \text{---} \end{array} \\ \theta &= 10^{-3} \end{aligned}$$


Transcoders (which reconstruct the output of an MLP from its input, using a sparse hidden layer) do not work that well.

Jumping ahead: improving reconstruction fidelity with JumpReLU sparse autoencoders
S. Rajamanoharan et al.

To train an SAE, with discontinuous JumpReLU activations and ℓ^0 sparsity penalty, use straight-through estimation for the gradients

$$\frac{\partial \text{JumpReLU}}{\partial \theta} * k_\varepsilon, \quad \frac{\partial \mathbf{1}_{z \geq \theta}}{\partial \theta} * k_\varepsilon$$

where k_ε is a (rectangle) kernel of bandwidth ε ; for the ℓ^0 penalty, use the pre-activations:

$$\|y\|_0 = \mathbf{1}_{W_1 x + b_1 \geq 0}.$$

Scaling and evaluating sparse auto-encoders
L. Gao et al.

The SAE with top- k activations ($k = 32$) does not require an ℓ^0 or ℓ^1 penalty, but may suffer from a large number of dead neurons: add an auxiliary loss, the reconstruction error from the top k_{aux} (512). You can use several values of k , e.g., $\ell(k) + \ell(4k)/8$, and change k at test time.

$$\begin{aligned} y &= \sigma(W_1(x - b_2) + b_1) \\ \hat{x} &= W_2 y + b_2 \end{aligned}$$

**Improving dictionary learning
with gated sparse autoencoders**
S. Rajamanoharan et al.

The L^1 penalty in SAE

$$\begin{aligned}x &= x - b_2 \\ y &= \text{ReLU}(W_1 x + b_1) \\ \hat{x} &= W_2 y + b_2\end{aligned}$$

causes shrinkage. Instead, *gated SAEs* use

$$\begin{aligned}x &= x - b_2 \\ g &= W_0 x + b_0 \\ y &= \mathbf{1}_{g \geq 0} \odot \text{ReLU}(W_1 x + b_1) \\ \hat{x} &= W_2 y + b_2 \\ (W_1)_{ij} &= e^{r_i} \cdot (W_0)_{ij}.\end{aligned}$$

**Transcoders find
interpretable LLM feature circuits**
J. Dunefsky et al.

Circuit analysis looks for sparse subgraphs corresponding to specific behaviours. Transcoders approximate a densely activating MLP layer with a wider, sparsely activating layer.

$$\begin{aligned}z &= \text{ReLU}(W_1 x + b_1) \\ \hat{y} &= W_2 z + b_2.\end{aligned}$$

Train with a faithfulness loss $\|\hat{y} - y\|^2$ and a sparsity penalty $\|z\|_1$.

To find interpretable circuits, pick the top k activations; among their children, pick the top k activations; among the resulting k^2 graphs, pick the top k ones; and continue.

**Sparse feature circuits: discovering and editing
interpretable causal graphs in language models**
S. Marks et al.

Neurons or attention heads are polysemantic. Instead, use sparse autoencoders and look for sparse feature circuits. After interpreting them, you may decide to remove some of them (e.g., gender, for fairness).

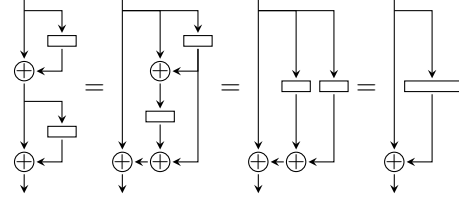
**Scaling monosemanticity: extracting
interpretable features from Claude 3 Sonnet**
A. Templeton et al.

Applications of sparse autoencoders (SAE) to interpret LLMs include identifying features activated when

- The model gives instructions to build chemical, biological or nuclear weapons;
- The user probes the model's goals and values, or tries to jailbreak it;
- The model is trained to be a sleeper agent;
- etc.

**Sparse cross-coders
for cross-layer features and model diffing**
J. Lindsey et al.

Adjacent layers are almost parallel: apply dictionary learning to them jointly (a cross-coder is a multi-layer transcoder).



Tokenizarion is NP-complete
P. Whittington et al.

The problem of compressing a dataset to at most δ symbols is NP-complete.

**Neural relational inference
for interacting systems**
T. Kipf et al. (2018)

GNN over a *latent* graph, to discover and model particle interactions (e.g., basketball players).

Can a transformer represent a Kalman filter?
G. Goel and P. Bartlett

Yes, explicitly, up to a small additive error.

**ZipIt! Merging models
from different tasks without training**
G. Stoica et al.

To merge models (same architecture but different training data and/or tasks), permutation-based methods assume they have exactly the same features, up to permutation, and try to recover that permutation, for each layer:

$$\text{Maximize}_{\sigma \in \mathfrak{S}_n} \sum_i \text{Cor}(z_i^1, z_{\sigma(i)}^2)$$

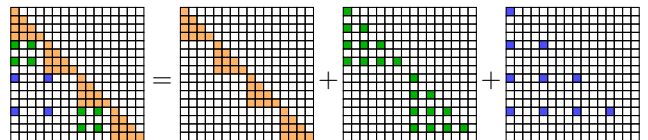
where z_i^2 are the features, for neuron i , in layer ℓ , which contains n neurons.

$$W_\ell^* = \frac{1}{2} W_\ell^1 + \frac{1}{2} P_\ell W_\ell^2 P_{\ell-1}^\top$$

Instead, concatenate the features, $z_i^1 \| z_i^2$, and average highly correlated ones: this accounts for redundancy in each model, and features unique to each model.

**LongNet:
scaling transformers to 1,000,000,000 tokens**
J. Ding et al.

Dilated attention is exponentially sparser as distance grows.



Consistency models
Y. Song et al.

The diffusion SDE

$$dx_t = \mu(x_t, t)dt + \sigma(t)dW_t$$

defines a probability flow ODE

$$\dot{x}_t = \mu(x_t, t) + \frac{1}{2}\sigma(t)^2\nabla\log p_t(x_t);$$

the score function $s_\theta(x, t) \approx \nabla\log p_t(x)$ is learned from data. Diffusion models solve this ODE, mapping $(x_{t+1}, t+1)$ to x_t . Consistency models instead directly map (x_t, t) to x_0 (or, rather, x_ε); they are trained from trajectories, and try to enforce the constraints

$$\begin{aligned} f(x_t, t) &= f(x_s, s) \\ f(x_\varepsilon, \varepsilon) &= x_\varepsilon. \end{aligned}$$

Diffusion models as masked autoencoders
C. Wei et al.

Diffusion model conditioned on masked input for image inpainting.

**Hyena hierarchy:
towards larger convolutional language models**
M. Poli et al.

Convolutions can be efficiently computed with the FFT

$$\begin{aligned} (h * u)_t &= \sum_{n=1}^{L-1} h_{t-n} u_n \\ (h * u) &= \begin{bmatrix} h_0 & h_{-1} & \cdots & h_{-L+1} \\ h_1 & h_0 & \cdots & h_{-L+2} \\ \vdots & & \ddots & \\ h_{L-1} & h_{L-2} & \cdots & h_0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_1 \\ \vdots \\ u_{L-1} \end{bmatrix}. \end{aligned}$$

The filter h can be implicit, $h_t = \gamma_\theta(t)$, where γ_θ is described by a neural net (with sine activations to capture frequency information, and possibly multiplied by a “shaping factor”, e.g., exponential decay) or comes from a state space model (SSM)

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t + Du_t \end{aligned}$$

i.e.,

$$\begin{aligned} y_t &= \sum_{n=0}^t (CA^{t-n}B + D\mathbf{1}_{t=n})u_n \\ h_t &= \begin{cases} XA^tB + D\mathbf{1}_{t=0} & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

where A, B, C, D are learned (the memory extent is determined by the spectral radius of A).

Hyena is a subquadratic replacement for self-attention: it first computes projections of the input (v, x^1, \dots, x^N)

(to mimick transformers, use 3, and call them value, key and query) and iterates

$$\begin{aligned} z_t^1 &= v_t \\ z_t^{n+1} &= x_t^n (h^n * z_t^n)_t \\ y_t &= z_t^{N+1} \end{aligned}$$

**Hungry hungry Hippos: towards language
modeling with state space models**
T. Dao et al.

The H3 state space model outputs

$$Q \odot \text{SSM}_{\text{diag}}(\text{SSM}_{\text{shift}}(k) \odot V)$$

where the shift SSL uses

$$A = \begin{bmatrix} 0 & & 0 \\ 1 & \diagdown & \\ & 1 & 0 \\ 0 & & 1 & 0 \end{bmatrix}$$

and the diagonal SSM uses a diagonal A .

**Designing losses for data-free training of
normalizing flows on Blotzman distributions**
L. Felardos et al.

Normalizing flows approximate a target distribution by learning a bijective transformation from some reference distribution (e.g., Gaussian).

$$\begin{array}{ccccc} \text{Gaussian} & q_N & \longrightarrow & p_G & \text{Generated} \\ & q_F & \longleftarrow & p_B & \text{Target} \end{array}$$

One can minimize $\text{KL}(p_G \| p_B)$: this only requires the unnormalized target density, but leads to mode collapse. Minimizing $\text{KL}(q_F \| q_N)$ requires data, which can be sparse.

Instead of data, one can use samples from the generated distribution, with importance sampling. For better convergence, the importance weights p_B/p_G should be close to 1.

As loss function, use

$$\text{Var}_{x \sim \text{generated}} \log \frac{p_B(x)}{p_G(x)}$$

or, more precisely,

$$\text{Mean}_i \left(r(x_{G,i}^\dagger) - k^\dagger \right)_+^2$$

where

$$\begin{aligned} r(x) &= \log \frac{p_B(x)}{p_G(x)} \\ k &= \text{Mean } r(x_{G,i}) \\ \dagger &= \text{stop-gradient.} \end{aligned}$$

**Action matching:
learning stochastic dynamics from samples**
K. Neklyudov et al.

If particles are described by the ODE $\dot{x}_t = v_t(x_t)$, their density satisfies $\dot{q}_t = -\nabla \cdot (q_t v_t)$. Under mild conditions, $v_t = \nabla s_t$ for some “action” s_t . It can be recovered by minimizing

$$\mathcal{L}_{\text{AM}}(s) = \mathbb{E}_{x \sim q_0} [s_0(x)] + \mathbb{E}_{x \sim q_1} [s_1(x)] + \int_0^1 \mathbb{E}_{x \sim q_t} \left[\frac{1}{2} \|\nabla s_t(x)\|^2 + \frac{\partial s_t}{\partial t} \right] dt$$

This can be generalized to SDEs (entropic action matching)

$$dx_t = v_t(x)dt + \sigma_t dW_t$$

and unbalanced action matching (creation or destruction of probability mass).

**The training process of many deep networks
explores the same low-dimension manifold**
J. Mao et al.

The *intensive PCA* (InPCA) of a distance matrix $D = (d(y_i, y_j))_{ij}$ is the eigendecomposition of the centered distance.

$$\begin{aligned} L &= I - \frac{1}{n} \mathbf{1} \\ W &= -\frac{1}{2} LDL \\ W &= U \Lambda U^\top \\ X &= U \sqrt{|\Lambda|} \end{aligned}$$

Coordinates corresponding to negative (resp. positive) eigenvalues have a negative (positive) contribution to distances.

**ZipLoRA: any subject in any style
by effectively merging LoRAs**
V. Shah et al

LoRA update matrices are sparse, but aligned LoRA matrices interfere when naively merged,

$$\Delta W = \lambda \Delta W_{\text{context}} + (1 - \lambda) \Delta W_{\text{style}}.$$

Instead, learn which column of which LoRA to keep,

$$(m'_1 \otimes \mathbf{1}) \odot \Delta W_1 + (m'_2 \otimes \mathbf{1}) \odot \Delta W_2.$$

Riemannian residual neural networks
I. Katsman et al. (2023)

Residual networks

$$x_{i+1} = x_i + \ell_{i+1}(x_i)$$

can be generalized to Riemannian manifolds

$$x_{i+1} = \exp_{x_i} \ell_{i+1}(x_i)$$

where the vector field ℓ_i is defined with a standard neural net

$$n_i : \mathbf{R}^D \longrightarrow \mathbf{R}^D$$

after embedding \mathcal{M} into a higher-dimensional vector space $\mathcal{M} \hookrightarrow \mathbf{R}^D$,

$$\ell_i(x) = \text{proj}_{T_x \mathcal{M}} n_i(x).$$

**RMT:
retentive networks meet vision transformers**
Q. Fan et al.

A retentive block is an attention block with exponential decay

$$\begin{aligned} \text{Ret}(X) &= (QK^\top \odot D)V \\ Q &= (XW_Q) \odot \Theta \\ K &= (XW_Q) \odot \bar{\Theta} \\ V &= XW_V \\ \Theta_n &= e^{in\theta} \\ D_{nm} &= \gamma^{n-m} \mathbf{1}_{n \geq m} \end{aligned}$$

It can be generalized to 2-dimensional data (Manhattan self-attention, MaSA).

Transformers from an optimization perspective
Y. Yang et al. (2022)

Unfolded optimization aims to replace the feed-forward layers of a neural network $y = f(x)$ with a gradient step from an optimization problem

$$\underset{y}{\text{Minimize}} \ g(y; x),$$

one layer at a time.

**Tokenformer: rethinking transformer scaling
with tokenized model parameters**
H. Wang et al.

Alternate standard attention

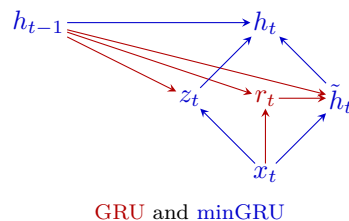
$$X \mapsto \text{Att}(XW_Q, XW_K, XW_V)$$

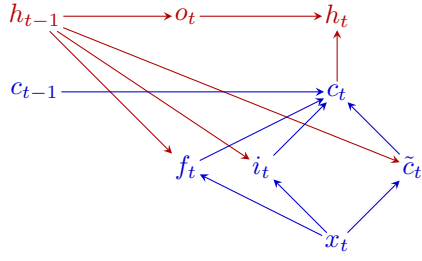
with token-parameter attention

$$X \mapsto \text{Att}(XW_Q, W_K, W_V).$$

Were RNNs all we needed?
L. Feng et al.

GRU and LSTM can be simplified, by removing the influence of the hidden state h_{t-1} on the gates.





LSTM and minLSTM

The gradients can be computed with the *parallel prefix scan* algorithm.

$$\begin{aligned} h_k &= a_k \odot h_{k-1} + b_k \\ \mathbf{a}^* &= \text{cumsum}(\log \mathbf{a}) \\ \mathbf{c} &= \log \text{cumsum} \exp(\log \mathbf{b} - \mathbf{a}^*) \\ \mathbf{h} &= \mathbf{a}^* + \mathbf{c} \end{aligned}$$

Efficient parallelization of a ubiquitous sequential computation F.A. Heinsen

The recurrence $x_t = a_t x_{t-1} + b_t$ can be computed with two cumulated sums (which can be computed in parallel),

$$\mathbf{x} = \left(\prod^{\text{cum}} a_t \right) \odot \left(x_0 + \sum^{\text{cum}} \frac{b_t}{\prod^{\text{cum}} a_t} \right),$$

i.e.

$$\begin{aligned} \log x_t &= a_t^* + \log(x_0 + b_t^*) \\ a_t^* &= \sum_t^{\text{cum}} \log a_t \\ b_t^* &= \sum_t^{\text{cum}} \exp(\log b_t - a_t^*) \end{aligned}$$

For numeric stability, prefer

$$x_t = \exp[a_t^* + \text{tail}(\text{LCSE}(\text{cat}(\log x_0, \log b_t - a_t^*)))].$$

Prefix sums and their applications G.E. Blelloch

Cumsum, cummax, cummin can be computed in parallel.

The recurrence $x_i = (x_{i-1} \otimes a_i) \oplus b_i$ can be computed in parallel.

$$\begin{aligned} (a, b) \cdot (a', b') &= (a \otimes a', b \otimes a' \oplus b') \\ \mathbf{x} &= \text{pr}_2(\text{cum}(\mathbf{a}, \mathbf{b})) \end{aligned}$$

This can be generalized to higher order recurrences (replace x_i with $(x_i, x_{i-1}, \dots, x_{i-k})$); applications include $x_i = a_i + b_i/x_{i-1}$, Fibonacci numbers, etc.

Byte latent transformer: patches scale better than tokens A. Pagnoni et al.

Train a small transformer to forecast the next byte, from the current and previous bytes, with *hash* n -gram embeddings, $3 \leq n \leq 8$. Use it to split the input into variable-length *patches* (whenever the entropy of the next byte is above some threshold or increases too much). Feed those patches to a large transformer, to forecast the next patch. Decode the patches into bytes, with a small transformer, still using the entropy to decide when patches end.

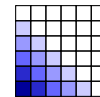
The model scales better than tokenizer-based transformers, and excels at character-level tasks.

Switch transformers: scaling to trillion parameter models with simple and efficient sparsity W. Fedus et al.

MoE (mixture of experts) selecting just one expert, with a load-balancing loss to use all experts.

Train short, test long: attention with linear biases enables input length extrapolation O. Press et al.

Position encoding (PE) adds a sine (or learned) positional encoding to the first layer. *RoPE* multiplies keys and queries with a sine encoding, at each layer; the values are left untouched. *T5 bias* adds a learned, distance-dependent bias to the pre-softmax attention, at each layer (the values are left as is). *ALiBi*



adds a penalty proportional to the distance to the pre-softmax attention.

$$\text{softmax} \left(q_i K_{:,i}^\top - m \begin{bmatrix} i-1 \\ \vdots \\ 2 \\ 1 \\ 0 \end{bmatrix} \right)$$

RoFormer: enhanced transformer with rotary position embedding J. Su et al.

xLSTM: extended long short-term memory M. Beck et al.

In the LSTM, replace

$$\begin{aligned} c_t &= \sigma(f_t) c_{t-1} + \sigma(i_t) z_t \\ h_t &= o_t \psi(c_t) \end{aligned}$$

with

$$\begin{aligned}c_t &= \exp(f_t)c_{t-1} + \exp(i_t)z_t \\h_t &= o_t \frac{c_t}{n_t} \\n_t &= \exp(f_t)n_{t-1} + \exp(i_t).\end{aligned}$$

To make it more transformer-like, replace the scalar c with a matrix, and use matrix products. (To avoid numeric overflow, keep track of $m = \log n_t$, and divide everything by e^m .)

***A Wigner-Eckart theorem
for group equivariant convolution kernels***
L. Lang and M. Weiler (2021)

The Peter-Weyl theorem states that if G is a compact group and X a homogeneous space, then $L^2(X, K)$ ($K = \mathbf{R}$ or \mathbf{C}) is a direct sum of unitary irreducible representations, each with multiplicity at most its dimension; for $X = G$ and $K = \mathbf{C}$, the multiplicity is the dimension.

***Coordinate-independent
convolutional networks***
M. Weiler et al.

***Nomic Embed: training a reproducible
long context text embedder***
Z. Nussbaum et al.

Open-source 8192-context-length embedding model, with *task-specific prefixes*, to break the symmetry of the encoder (we want a different model for questions and answers, such that the embeddings match). Tricks include: RoPE, SwiGLU, Flash Attention, AdamW, gradient accumulation, etc.

***Self-guiding exploration
for combinatorial problems***
Z. Iklassev et al.

To solve combinatorial problems (e.g., TSP) with an LLM:

- “List all possible methods to solve this problem. Return them separated by newlines”;
- For each method, “list all the steps to use this method”;
- For each step, “Is the problem easily solvable? Return ‘yes’ or ‘no’”; if not, go to the previous step;
- “Give feedback on the proposed solution”;
- “Integrate all previous findings and provide the final answer”.

Stealing part of a production language model
N. Carlini et al.

With access to the logits of an LLM, one can recover the dimension of the last weight matrix. The final layer is a projection from the hidden latent dimension to a *higher* dimensional logit vector: the dimension of the

span of the logit vectors is the dimension of the last layer; estimate it by looking at the singular value of the matrix Q of logits. The SVD, $Q = U\Sigma V^\top$ also gives the weight matrix, $W = U\Sigma$ (up to symmetries). The attack still works without direct access to the logits, but is more expensive.

***Let’s do a thought experiment: using
counterfactuals to improve moral reasoning***
X. Ma et al. (2023)

To help LLMs to better in moral reasoning tasks:

- Ask for moral counterfactual questions for each scenario;
- Answer; discuss moral implications; highlight moral conflicts;
- Summarize and conclude.

***Tree of thoughts: deliberate problem solving
with large language models***
S. Yao et al.

Don’t ask for the next step, as in CoT (chain of thoughts), but for several possible next steps; explore the tree with BFS (ask the LLM to pick the best k children among those generated; some tasks (e.g., creative writing) are fine with $k = 1$).

***ChatDB: augmenting LLMs
with databases as their symbolic memory***
C. Hu et al.

Give your LLM access to an SQL database, where it can store (and later retrieve) information.

Learning to tokenize for general retrieval
W. Sun et al.

Sparse retrieval relies on inverted indices (TF-IDF, BM25, etc.). Dense retrieval compares queries and document embeddings. Generative retrieval replaces the dense embeddings with discrete ones – lists of “tokens”.

***Retrieval augmented generation
for large language models: a survey***
Y. Gao et al. (2023)

RAG can suffer from low precision, low recall, irrelevance, hallucinations:

- Preprocess the chunks: remove irrelevant or duplicate information (e.g., boilerplate text, special characters); rewrite to eliminate ambiguity; adjust chunk size;
- Align query and documents: given a document, ask for corresponding queries and use them to compute the embedding; conversely, given a query, ask for (hallucinated) documents answering it, and query the vector database with them;
- Add metadata to the retrieved documents;
- Use a customized (domain-specific) embedding;
- Re-rank;

- Reduce noise in the retrieved documents and compress them (reduce the context length): remove irrelevant context, highlight pivotal paragraphs; ask the LLM to rewrite the context;
- Hybrid search (vector + keywords);
- Recursive retrieval: first retrieve small chunks, to capture key information, then larger blocks for more context;
- Ask the LLM to “step back and reason about the general underlying concepts or principles”;
- Let the LLM populate its own database;
- Ask the LLM to assess the relevance of the retrieved documents;
- Rewrite the query; try several formulations;
- Compute the embeddings on small chunks, but retrieve larger chunks (e.g., abstracts vs full papers);
- etc.

Time is encoded in the weights of finetuned language models
K. Nylund et al.

The time vectors are $W_t - W$, where W are the model weights and W_t the weights after finetuning on data from period t . Use them to shift the model’s attention to another period, e.g., $W + (W_{t_1} - W_{t_0})$.

“Do anything now”: characterizing and evaluating in-the-wild jailbreak prompts on large language models
X. Shen et al. et al. (2023)

Get your jailbreak prompts from Reddit and Discord.

Patch n’ Pack: NaBiT, a vision transformer for any aspect ratio and resolution
M. Dehghani et al.

Instead of resizing images before giving them to a ViT, pack patches from several images, and have the network output a sequence of labels, one for each input image.

Domain adaptation for time series under feature and label shifts
H. He et al.

Align the two domains (feature shift: time and frequency features, Sinkhorn divergence), then correct (label shift).

Baldur: whole-proof generation and repair with large language models
E. First et al.

LLM to generate formal proofs (software verification with Isabelle) and fix them in case of errors.

Prompt programming for large language models: beyond the few-shot paradigm
L. Reynolds and K. McDonell

Few shot prompts help locate already learnt tasks: well-crafted 0-shot prompts can do better.

Safety alignment should be made more than just a few tokens deep
X. Qi et al.

Safety alignment of LLMs with RLHF is shallow: it just trains the model to identify harmful requests and to *start* the answer with a refusal (“I cannot”, etc.). Those models are susceptible to *prefilling attacks* (provide the beginning of the answers, e.g., “Sure, here is” and let the model complete it).

JaxPruner: a concise library for sparsity research
J.H. Lee et al. (2024)

Implementation of pruning and sparse training algorithms.

A* search without expansions: learning heuristic functions with deep Q-networks
F. Agostinelli et al.

Deep Q network as heuristic for A* search (aka best-first search).

Dynamic clustering of contextual multi-armed bandits
T.T. Nguyen and H.W. Lauw (2014)

DynUCB:

- Fits a LinUCB model for each user;
- Clusters the coefficients of those models (k -means);
- Selects the action using the average model in each cluster.

Nonparametric bandits with covariates
P. Rigole and A. Zeevi (2018)

For the (non-contextual) 2-arm bandit, UCB is asymptotically optimal and its regret is $O(\log n)$. For a contextual 2-arm bandit, bin the context vectors (using a grid) and use UCB on each cell; the regret is polynomial – there exist bandits for which this cannot be improved.

Nonparametric stochastic contextual bandits
M.Y. Guan and H. Jiang (2018)

(Under reasonable assumptions), k -NN-UCB has sub-linear regret.

A practical method for solving contextual bandit problems using decision trees
A.N. Elmachetoub et al.

For each possible action, fit a decision tree on a bootstrap sample, and use it as a Thompson sample: predict the reward, and pick the action with the highest reward.

***The epoch-greedy algorithm
for contextual multi-armed bandits***
J. Langford and T. Zhang

Iterate:

- One exploration step (random arm);
- $\lceil \varepsilon_n \rceil$ exploitation steps, where ε_n is an average regret bound for each exploration step, for the policy optimal (from a finite set \mathcal{H} of candidate policies) on the exploration data only.

***Gambling in a rigged casino:
the adversarial multi-armed bandit problem***
P Auer et al. (1998)

Exp3 solves the fixed-horizon multi-arm bandit (MAB) problem with k arms as

$$p_i(t) \propto \exp \eta \sum_{\tau < t} \hat{r}_{i\tau}$$

$$\hat{r}_{it} = \begin{cases} \frac{r_{it}}{\hat{p}_{it}} & \text{if arm } i \text{ was chosen at time } t \\ 0 & \text{otherwise} \end{cases}$$

$$\hat{p}_{it} = (1 - \varepsilon)p_{it} + \frac{\varepsilon}{k}$$

Exp4 uses the same idea to choose between experts (each providing a probability distribution on arms).

***Approximate planning in large POMDPs
via reusable trajectories***
M. Kearns et al.

Given a POMDP you can sample from, with a finite action space, build *trajectory trees*:

- Nodes: state, observation pairs
- Children for all actions, with the edge labeled by the corresponding reward;
- Depth H (there is an exponential number of nodes in each tree).

Use those trees to evaluate policies: a deterministic (resp. stochastic) policy defines a (distribution on) path(s).

Sequential testing
R. Nowak (2011)

Consider the null and alternative hypotheses

$$H_0 : X_i \stackrel{\text{iid}}{\sim} p_0$$

$$H_1 : X_i \sim p_1$$

(they have to be simple); choose α and β , the desired type I and type II error rates; compute the thresholds

$$a = \log \frac{\beta}{1 - \alpha} < 0$$

$$b = \log \frac{1 - \beta}{\alpha} > 0$$

and the cumulated sums

$$\Lambda_k = \sum_{i=1}^k \log \frac{p_1(x_i)}{p_0(x_i)}.$$

The *sequential probability ratio test* (SRPT) accepts H_p (resp H_0) as soon as $\Lambda_k > b$ (resp $\Lambda_k < a$).

***Doubly robust policy evaluation
and optimization***
M. Dudík et al. (2014)

The *direct method* (DM, aka controlling for confounders, backdoor adjustment) estimates the value of a (MAB) policy from an estimate $\hat{r}(s, a)$ of the reward conditioned on the state (state = context = exogenous variables) s and action a

$$V = \text{Mean}_t \sum_a \pi(a, s_t) \hat{r}(s_t, a);$$

it has low variance but potentially high bias.

Importance sampling (IS) uses the (known) data collection policy π_0

$$V = \text{Mean}_t \frac{\pi(a_t, s_t)}{\pi_0(a_t, s_t)} r_t$$

(there is no bias, but the variance is high if π and π_0 are very different).

The *inverse propensity* score (IPS, or inverse propensity weighting, IPW) method uses some estimator $\hat{\pi}_0$ of the collection policy,

$$V = \text{Mean}_t \frac{\pi(a_t, s_t)}{\hat{\pi}_0(a_t, s_t)} r_t.$$

The *doubly robust* (DR) estimator uses \hat{r} (DM) as a baseline, and corrects it with the propensity score

$$V = \text{Mean}_t \sum_a \pi(a, s_t) \hat{r}(s_t, a) + \frac{\pi(a_t, s_t)}{\hat{\pi}_0(a_t, s_t)} (r_t - \hat{r}(s_t, a_t)).$$

***Open bandit dataset and pipeline: towards
realistic and reproducible off-policy valuation***
Y. Saito et al. (2021)

Besides DM, IPW, DR, *off-policy evaluation* (OPE) of contextual MAB policies include

- Self-normalized IPW

$$V = \frac{\text{Mean}_t w(a_t, s_t) r_t}{\text{Mean}_t w(a_t, s_t)}, \quad w(a, s) = \frac{\pi(a, s)}{\hat{\pi}_0(a, s)}$$

- Switch estimator: DR when $w \leq \tau$, DM when the importance weights are too large;
- DR with optimistic shrinkage

$$V = \text{Mean}_t \hat{r}(s_t, \pi) + \hat{w}(a_t, s_t, \lambda) [r_t - \hat{r}(s_t, a_t)]$$

$$\hat{w}(a, s, \lambda) = \frac{\lambda}{w(a, s) + \lambda} w(a, s)$$

- Cross-fitting.

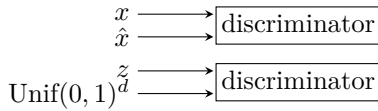
***TimesNet: temporal 2D-variation modeling
for general time series analysis***
H. Xu et al.

To account for inter- and intra-period variations, and multi-periodicity, stack (with residual connection) several TimesBlocks:

- Compute the FFT of the 1D input (if there are several channels, average the amplitudes of each frequency) to identify the top k frequencies
- For each of those frequencies, reshape the 1D input to 2D (*i.e.*, stack them), with 0-padding if needed;
- Apply an inception block;
- Reshape to 1D.

***Generative probabilistic forecasting
with applications in market operations***
X. Wang and L. Tong.

A weak autoencoder only recovers its input in distribution.



***Generative time series modeling
with Fourier flows***
A.M. Alaa et al.

Normalizing flows in the frequency domain

***Predict, refine, synthesize:
self-guiding diffusion models
for probabilistic time series forecasting***
M. Kollovieh et al.

TSDiff is an unconditional diffusion model for time series with S4 layers. To make it conditional, *diffusion guidance* uses Bayes rule

$$\nabla_{x_t} \log p(x_t|c) = \nabla_{x_t} \log p(x_t) + \nabla_{x_t} \log p(c|x_t)$$

which only requires an auxiliary classifier (for forecasting, $c = x^{\text{obs}}$).

The *continuous ranked probability score* (CRPS) is the quantile (pinball) loss integrated from 0 to 1; it can be used to evaluate probabilistic forecasts.

***Score matching through the roof:
linear, nonlinear, and latent variables
causal discovery***
F. Montagna et al.

The score function $\nabla \log p$ can be used to test for conditional independence:

$$X \perp\!\!\!\perp Y | Z \iff \frac{\nabla^2}{\nabla X \nabla Y} \log p(X, Y, Z) = 0.$$

This leads to a generalization of the PC and PCI algorithms: AdaScore.

***Sensitivity analyses
for unmeasured confounders***
L. D’Agostino McGowan (2022)

Sensitivity analysis answers the question “how strong should a missing confounder be to change the sign of $E[Y|\text{do}(T=1)]?$ ”

R implementation in `tipr`.

***Learning domain-specific
causal discovery from time series***
X. Wang and K. Kording

Supervised learning for causal discovery (for time series, with a transformer).

***Granger causality detection with
Kolmogorov-Arnold networks***
H. Lin et al.

Granger causality testing with KAN (stacked GAMs), with a sparsity penalty for the first layer, fitted with proximal gradient descent to ensure sparsity.

KAN: Kolmogorov-Arnold networks
Z. Liu et al.

Stacked GAMs.

KAN: Kolmogorov-Arnold networks
Z. Liu et al.

A KAN is a neural net with GAM layers: the weights are replaced by learnable nonlinearities (splines of order 3 on G intervals).

***NeuralFactors: a novel factor learning
approach to generative modeling of equities***
A. Gopal

Train a VAE to forecast future stock returns from current stock characteristics

$$\begin{aligned} \text{stock}_{it} &\mapsto \alpha_{it}, \beta_{it}, \sigma_{it}, \nu_{it} \\ z_{t+1} &\sim T(\mu_z, \sigma_z, \nu_z) \\ r_{i,t+1} &\sim T(\alpha_{it} + \beta_{it}^\top z_{t+1}, \sigma_{it}, \nu_{it}) \end{aligned}$$

and use it for synthetic data, covariance estimation, risk analysis, portfolio construction.

A T-SNE embedding of the β_i reveals industrial sectors.

Improving investment strategies with GNNs
G. Farmanfarmaian

Gated GCNs use GRU in the depth dimension.

GraphGPS alternates MPNN layers with global attention layers.

The *turbulence index* (of the market, at a given data) is the Mahalanobis distance to the average return vector.

The structure of the supply chain graph is not useful in forecasting future (monthly) returns, but global information (e.g., from a virtual node) is.

***A geometric approach
to asset allocation with investor views***
A.V. Antonov et al. (2024)

Given a prior on the distribution of asset returns

$$x \sim p_{\text{prior}} = N(\mu_0, V_0)$$

and investor views $x \sim p_{\text{views}} = N(\mu_1, V_1)$, compute a posterior distribution

$$\underset{p=N(\mu, V)}{\text{Argmin}} d(p, p_{\text{prior}}) \text{ such that } d(p, p_{\text{view}}) \leq d_0$$

or

$$\underset{p=N(\mu, V)}{\text{Argmin}} d(p, p_{\text{prior}}) + \lambda d(p, p_{\text{view}}) \leq d_0.$$

This readily generalizes to degenerate views.

Markowitz portfolio construction at seventy
S. Boyd et al. (2024)

Long list of constraints and penalties you may want to add to your portfolio optimization (turnover, risk budget, costs, etc.).

Replace the hard constraints with penalties to ensure the problem is always feasible (but report constraint violations); fine-tune the scale of the penalties on historical data.

Use robust optimization to account for the uncertainty in μ and Σ .

Some constraints are not convex:

- Integral number of shares (lot shares);
- Maximum number of assets;
- Minimum position size;

Do not write the target volatility constraint as $w' \Sigma w \leq \sigma^2$, but $\|L^\top w\|_2 \leq \sigma$, where $\Sigma = L^\top L$ is the Choleski decomposition. If Σ comes from a factor model, $\Sigma = FvF^\top + \Delta$, use

$$\begin{aligned} w' \Sigma w &= \|\ell^\top w\|_2^2 + \|\Delta^{1/2} w\|_2^2 \\ &= \left\| \begin{pmatrix} \ell^\top w \\ \Delta^{1/2} w \end{pmatrix} \right\|_2^2 \end{aligned}$$

where $v = \ell^\top \ell$.

***Constrained max drawdown: a fast and robust
portfolio optimization approach***
A. Dorador

The Markowitz portfolio optimization problem (minimizing risk with a minimum return constraint) can be linearized: replace the risk with

$$\sum_t \left| \sum_i (r_{it} - \bar{r}_i) w_i \right|$$

or

$$\text{Min}_t \sum_i r_{it} w_i$$

(add $0 \leq w \leq \frac{1}{2}$ to avoid the 1-asset portfolio). To limit the number of assets (e.g., $\forall i w_i \neq 0 \implies w_i \leq 0.05$), use the big-M transform (with $M = 0.5$).

***Can GANs learn the stylized facts
of financial time series***
S. Kwon and Y. Lee

GANs struggle to reproduce simple statistical models (Heston, Ornstein-Uhlenbeck, jump diffusion, etc.)

Time-series foundation model for value-at-risk
A. Goel et al. (2024)

Fine-tuned foundation time series models (e.g. TimeFM) are competitive with conventional econometric models (GARCH, GAS).

***Automated regime detection
in multidimensional time series data
using sliced Wasserstein k-means clustering***
Q. Luan and J. Hamp (2023)

Sliced Wasserstein k -means clustering on (the delay emneddings of) time series.

***Risk parity portfolios with skewness risk:
an application to factor investing
and alternative risk premia***
B. Bruder et al. (2016)

Define risk parity portfolios using skewness as a risk measure, and assuming asset returns follow a Gaussian mixture distribution.

A tale of tail covariances (and diversified tails)
J. Rosensweig (2023)

The *tail covariances* are $E[XY^{2k-1}]$ and $E[X^{2k-1}Y]$ (higher moments depend more on the tail of the distributions than their centers).

***Multistage stochastic programs
with the entropic risk measure***
O. Dowson et al.

$$\text{Ent}_\gamma[X] = \gamma^{-1} \log E[e^{\gamma X}]$$

***Industry classification based on supply chain
network information using GNNs***
D. Wu et al.

Infer the industry of unlisted firms from the supply chain network.

The corporate chronicles
Wolfe research

How markets react to corporate events (40+ types: earnings, clients, products, corporate structure, listing, dividends, buybacks, legal, bankruptcy, regulations, shareholder activism, etc.)

Factor crowding, tail dependence, and dynamic allocation
Wolfe research (2021)

Define crowding as

Utilization(short leg) – Utilization(long leg).

More generally, regress utilization against factor quantile, size quantile and volatility quantile; use the coefficients of the factor quantiles as crowding for each quantile.

Measuring stock-level exposure to bitcoin
Wolfe research (2021)

Use the grayscale bitcoin trust (GBTC) instead of bitcoin, to have market-like closing prices. Find stocks exposed to bitcoin in two ways:

- Exposure: return \sim market + industry + BTC, with Huber loss, and elastic net penalty
- NLP: cryptocurrency related words in management presentations and conference calls.

Text-based network industries and endogenous product differentiation
G. Hoberg and G. Phillips (2009)

Take the product description from the 10K filings, only keep nouns and proper nouns (they are capitalized 90% of the time), discard geographic words, and words appearing in more than 25% of the documents; compute pairwise cosine similarity to get each company's neighbours.

Either use this weighted graph to compute a fixed industry classification (apply some clustering algorithm on data from 1997) or truncate the graph (to have the same number of connected pairs as in the SIC-3 classification).

Deep generators on commodity markets; application to deep hedging
N. Bourson et al.

Synthetic time series: TSGAN, CTOGAN, SIGGAN, CEGEN.

Higher-order graph attention network for stock selection with joint analysis
Y. Qiao et al.

Stack LSTM (stock characteristics), motif-based GAT (from a graph built from sector, industry, Wiki data, etc.), MLP.

Simple formulas for standard errors that cluster by both firm and time
S.B. Thompson (2009)

In a panel regression $y_{it} = x'_{it}\beta + \varepsilon_{it}$, if there is correlation among i, t and (i, t) (up to lag L)

$$E[\varepsilon_{it}\varepsilon_{is}|x_{it}x_{is}] \neq 0$$

$$E[\varepsilon_{it}\varepsilon_{jt}|x_{it}x_{jt}] \neq 0$$

$$E[\varepsilon_{it}\varepsilon_{js}|x_{it}x_{js}] \neq 0 \text{ if } |t - s| < L$$

then

$$\text{Var } \hat{\beta} = \hat{V}_I + \hat{V}_{T,0} - \hat{V}_{W,0} + \sum_{1 \leq \ell \leq L} (V_{T,\ell} + V'_{T,\ell}) - \sum_{1 \leq \ell \leq L} (\hat{V}_{w,\ell} + \hat{V}'_{w,\ell})$$

where

$$\hat{V}_I = H^{-1} \sum_i \hat{c}_i \hat{c}'_i H^{-1}$$

$$\hat{V}_{T,\ell} = H^{-1} \sum_t \hat{s}_t \hat{s}'_{t+\ell} H^{-1}$$

$$\hat{V}_{w,\ell} = H^{-1} \sum_{it} \hat{u}_{it} \hat{u}'_{i,t+\ell} H^{-1}$$

$$\hat{u}_{it} = x_{it} \hat{\varepsilon}_{it}$$

$$\hat{c}_i = \sum_t \hat{u}_{it}$$

$$\hat{s}_t = \sum_i \hat{u}_{it}$$

$$\hat{\varepsilon}_{it} = y_{it} - x_{it} \hat{\beta}$$

$$H = \sum_{it} x_{it} x'_{it}$$

Estimating standard errors in finance panel data sets: comparing approaches
M.A. Petersen (2006)

Commonly used estimators of the standard error in panel regressions are biased:

- Newey-West has a small bias;
- Fama-MacBeth only accounts for correlation between firms;
- Clustered standard errors only account for correlation over time.

The square-and-add Markov chain
P. Diaconis et al.

Adding $\pm 1 \bmod p$ is a random walk on \mathbf{F}_p , which converges (slowly) to a uniform distribution; $X_{n+1} = 2X_n \pm 1 \bmod p$ converges faster; $X_{n+1} = X_n^2 \pm 1$ (in \mathbf{F}_q) looks trickier.

2. Classical portfolio optimization methods include Markowitz, risk parity, and hierarchical risk parity.

3. The first chapters cover classical topics:

- Mean-variance optimization with or without a risk-free asset; tangent portfolio;
- 2-fund separation theorem;
- CAPM;
- Multifactor model (APT);
- Shrinkage or random matrix theory (RMT) to estimate the variance matrix;
- Returns estimation: historical, Black-Litterman, Jackknife (bias correction), penalized regression, bootstrap (for confidence intervals);
- Portfolio optimization with CVaR constraint

and Bayesian methods

- Black-Litterman;
- gfbm (geometric fractional Brownian motion): geometric Brownian motion with the Wiener process replaced with a fractional Brownian motion W^H , with Hurst exponent H , defined as a Gaussian process (GP) with covariance function

$$E[W_t^H W_s^H] = \frac{1}{2}(|t|^{2H} + |s|^{2H} - |t - s|^{2H}),$$

with a beta prior on the Hurst exponent H

- Metropolis-Hastings
- Kalman filter, particle filter
- Hierarchical Bayesian models
- Bayesian optimization, GP regression, GP classification

5. Besides VaR and CVaR (technically, tail conditional expectation, expected shortfall and conditional value at risk are different, but they coincide if the distribution of X is smooth and $E[X^-] < \infty$),

$$\text{TCE}_\alpha = E[Y | Y \geq \text{VaR}_\alpha(Y)]$$

$$\text{ES}_\alpha = \frac{1}{1 - \alpha} \int_\alpha^1 \text{VaR}_u(Y) du$$

$$\text{CVaR}_\alpha = \inf_{a \in \mathbf{R}} a + \frac{E[(Y - a)^+]}{1 - \alpha}$$

also consider

- Gini shortfall

$$\rho(Y) = E[Y_\alpha] + \beta E[|Y'_\alpha - Y''_\alpha|], \quad \beta < \frac{1}{2}$$

where $Y_\alpha = Y | Y \leq \text{VaR}_\alpha$ and Y'_α and Y''_α are iid copies of Y_α ;

- Spectral risk measures,

$$- \int_0^1 q_X(u) \phi(u) du,$$

with $\phi \geq 0$, $\phi \searrow$, $\int |\phi| = 1$, e.g., the exponential spectral risk measure, with

$$\phi(u) = \frac{ke^{-ku}}{1 - e^{-k}}.$$

6. Factor models can be:

- Statistical (PCA-like);
- Macroeconomic (time series);
- cross-sectional (à la Fama-French).

This chapter also contains a proof of a (rigorous version of) the fundamental law of portfolio management.

8. The *optimal control* problem

$$\begin{aligned} &\text{Find} && u \\ &\text{To maximize} && \int_0^T F(x_t, u_t, t) dt + g(x_T) \\ &\text{Such that} && \dot{x} = f(x_t, u_t, t), x(0) = x_0 \end{aligned}$$

leads to the Hamilton-Jacobi-Bellman (HJB) equation

$$-\frac{\partial V}{\partial t} = \max_u F(x, u, t) + \nabla_x V \cdot f(x, u, t)$$

where $V(x_t, t) = \int_t^T F dt + g$.

A **viscosity solution** V of $F(x, V(x), \nabla V(x)) = 0$, $x \in \Omega$ is a function V satisfying

- (i) For all $\phi \in \mathcal{C}^1(\Omega)$, if $V - \phi$ has a local maximum at x_0 , then $F(x_0, V(x_0), \nabla \phi(x_0)) \leq 0$;
- (ii) For all $\phi \in \mathcal{C}^1(\Omega)$, if $V - \phi$ has a local minimum at x_0 , then $F(x_0, V(x_0), \nabla \phi(x_0)) \geq 0$.

The *Merton problem* looks for the proportion π of wealth to invest in a risky asset (assuming a geometric Brownian motion, with drift, volatility and risk-free rate known) to maximize expected CRRA utility

$$\begin{aligned} U(x) &= \frac{x^{1-\gamma}}{1-\gamma} \\ dX &= rX dt + (\mu - r)\pi X dt + \sigma \pi X dW; \end{aligned}$$

it is constant (it does not depend on time and wealth):

$$\pi^* = \frac{\mu - r}{\gamma \sigma^2}.$$

One can add transaction costs, or change the utility function.

9. Applications of Markov decision processes (MDP) include

- Multi-period portfolio optimization, with transaction costs or regime switching;
- Option hedging.

To make MDPs aware of risk, one could alter the reward; alternatively, one could directly alter the Bellman equation, either replacing the expected reward with the expected utility of reward, or by adding a variance penalty (risk-sensitive MDPs).

10. Optimal control and reinforcement learning (RL) study the same problem (OC tends to use continuous state and time, while RL prefers discrete time).

$$\dots s_t \xrightarrow{\text{agent}} a_t \xrightarrow{\text{environment}} s_{t+1}, r_{t+1} \xrightarrow{\text{agent}} a_{t+1} \dots$$

A RL model keeps estimates of one or more of the following:

- Policy $\pi(a|s)$;
- Value function $V(s)$;
- Action value function $Q(s, a)$;
- Model $\hat{p}(s'|s, a)$, $\hat{r}(s, a, s')$.

11. *Physics-informed neural nets* can fit a PDE $Df_\theta = 0$ (Black-Scholes or, more generally, the Fokker-Planck PDE of some SDE – the shape of the PDE is known, but some of its parameters are not) to data:

$$\text{Minimize}_{\theta} \text{Mean}_i \|f_\theta(x_i, t_i) - u_i\|^2 + \text{Mean}_j \|Df_\theta(x_j, t_j)\|^2.$$

They can also be used for SDEs, $dX = \mu dt + \sigma dW$, by minimizing

$$\left(\frac{dX}{dt} - \mu\right)^2 \quad \text{and} \quad [(X_{t+\Delta t} - X_t)^2 - \sigma^2 \Delta t]^2;$$

one can also add jumps.

12. Build the MST or TMFG of the correlation matrix of the asset returns, and invest in low-centrality assets (on average), while avoiding investing in nearly assets.

13. Hierarchical risk parity can be applied to other matrices, e.g.:

- Compute the sensitivity B of asset returns \mathbf{r} to risk factors \mathbf{f} : $\mathbf{r} = B\mathbf{f} + \varepsilon$ (with a linear model, or a sparse neural net – use automatic differentiation to compute B);
- Compute the distances between the assets in sensitivity space;
- Use that distance matrix to build a hierarchical risk parity portfolio.

Portfolio optimization based on neural network sensitivities from assets dynamics respect common drivers
A.R. Dominguez

Relevance
M. Czaronis et al.

The OLS forecast can be written as a “relevance-weighted” average of past observations,

$$\hat{y} = \bar{y} + \frac{1}{N-1} \sum_{i=1}^N r_{it}(y_i - \bar{y})$$

where the relevance between observations is

$$r_{ij} = \text{similarity}_{ij} + \frac{1}{2}(\text{informativeness}_i + \text{informativeness}_j)$$

$$\text{similarity}_{ij} = -\frac{1}{2}(x_i - x_j)'V^{-1}(x_i - x_j)$$

$$\text{informativeness}_i = (x_i - \bar{x})'V^{-1}(x_i - \bar{x}).$$

If you only use the subset $I \subset \llbracket 1, N \rrbracket$ of most relevant observations, use

$$\hat{y} = \bar{y} + \frac{\lambda^2}{|I|-1} \sum_{i \in I} r_{it}(y_i - \bar{y})$$

$$\lambda^2 = \frac{1}{N+1} \sum_{i=1}^N r_{it}^2 \Big/ \frac{1}{|I|+1} \sum_{i \in I} r_{it}^2.$$

Relevance-based prediction: a transparent and adaptive alternative to machine learning
M. Czaronis et al.

Relevance-based prediction

$$\begin{aligned} \hat{y} &= x'_{\text{new}} \hat{\beta} \\ &= \bar{y} + \frac{1}{N-1} \sum r_i (y_i - \bar{y}) \\ &= \sum w_i y_i \end{aligned}$$

expresses OLS forecasts as (data-dependent) linear combinations of observations

$$\begin{aligned} r_i &= \text{sim}(x_{\text{new}}, x_i) + \frac{1}{2} [\text{info}(x_{\text{new}}) + \text{info}(x_i)] \\ \text{sim}(x_1, x_2) &= -\frac{1}{2}(x_1 - x_2)'V^{-1}(x_1 - x_2) \\ \text{info}(x) &= (x - \bar{x})'V^{-1}(x - \bar{x}). \end{aligned}$$

We can also recover R^2 as

$$\begin{aligned} R^2 &= \frac{1}{N-1} \sum \text{info}(x_i) \text{fit}_i \\ \text{fit}_i &= \text{Cor}(w, y)^2 \end{aligned}$$

The virtue of transparency: how to maximize the utility of data without overfitting
M. Czaronis et al. (2024)

Use partial sample regression (relevance-based prediction on a subset of high-relevance observations) on a grid, with various relevance thresholds r^* (deciles) and all possible subsets of variables; use the adjusted fit as weights to combine the forecasts (as in BMA, Bayesian model averaging)

$$\begin{aligned} \hat{y} &= \sum w_i y_i \\ \text{fit} &= \text{Cor}(w, y) \\ \text{asymmetry} &= \frac{1}{2} [\text{Cor}(w^+, y) - \text{Cor}(w^-, y)]^2 \\ \text{adjusted fit} &= \# \text{variables} \times (\text{fit} + \text{asymmetry}) \end{aligned}$$

where w^+ and w_- are the weights if we use the observations with $i \geq r^*$ and $r_i < r^*$ (so that $w^+ = w$).

A transparent alternative to neural networks with an application to predicting volatility
M. Czaronis et al. (2024)

Relevance grid numeric example: predicting volatility from 14 variables (do not use all the subsets of predictors, just 100 random ones).

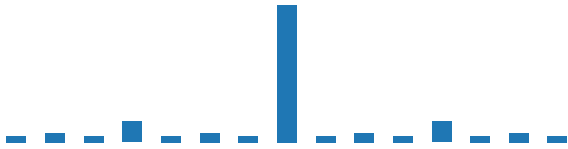
Any deep ReLU network is shallow
M.J. Villani and N. Schoots

A ReLU network is a locally linear function on a partition into polytopes. Any such function can be represented by a ReLU network, and 3 hidden layers are enough. This transformation can be automated, and the resulting network may be easier to interpret (Shapley, etc.)

Provably faster gradient descent via long steps

B. Grimmer

Gradient descent, with periodic long steps (which may increase the loss function), speeds up convergence (from $1/2T$ to $1/5T$)



Can large language models infer causation from correlation

Z. Jin et al.

Ask an LLM to discover and run the PC algorithm, *i.e.*, discover and use

- $A \perp\!\!\!\perp B | C \implies$ no A - B edge;
- $A \perp\!\!\!\perp B$ and $A \not\perp\!\!\!\perp B | C \implies A \overset{A}{\underset{B}{\rightrightarrows}} C$;
- All colliders can be found in that way.

(given all the relevant conditional independence statements, with no noise). The performance is bad.

Full parameter fine-tuning for large language models with limited resources

K. Lv et al.

To reduce memory usage when training large models, replace the traditional

- Forward pass, keeping all the activations;
- Backward pass, keeping all the gradients;
- Parameter update

with

- Forward pass
- For each layer, starting from the last:
 - Compute the gradients;
 - Update the parameters;
 - Free all the layers after the current one (they are no longer needed).

Language models can solve computer tasks

G. Kim et al. (2023)

When asking an LLM to generate keyboard and mouse actions, iterate:

- “Review your previous answer and find problems with it”;
- “Based on the problems found, improve your answer”.

[This is similar to LLM-based optimization.]

Deep symbolic regression for physics guided by unit constraints: towards the automated discovery of physical laws

W. Tenach et al. (2023)

Physical-unit-aware symbolic regression, with deep reinforcement learning, for low-noise data (0.1% to 10%): dimensional analysis drastically reduces the search space.

Language model cross-over: variation through few-shot prompting

E. Leyerson et al.

CMA-ES on sentences, with an LLM to do the sampling. For instance,

- For symbolic regression, try “here are 10 expressions that approximate the dataset”, followed by 9 lines of Python code;
- Stable diffusion prompts, to generate images (the objective function could be the dominating colour, or anything that can be computed automatically);
- Modifying the sentiment of a sentence (without changing its meaning);
- Reinforcement learning (generate Python code to control a robot).

Inference-time intervention: eliciting truthful answers from a language model

K. Li et al. (2023)

Find a “truthfulness” direction, for some attention heads (with a probe, *i.e.*, a classifier trained on the network activations), and shift activations in that direction at inference time.

System 2 attention (is something you might need too)

J. Weston and S. Sukhbaatar

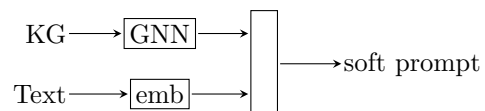
Instead of giving all the context to an LLM, proceed in two steps:

- Ask the LLM to remove irrelevant parts of the context;
- Answer the query.

Graph neural prompting with large language models

Y Tian et al.

Combine knowledge graph and LLMs with soft prompts.



LLaMA-adaptor: efficient fine-tuning of language models with zero-init attention

R. Zhang et al.

Fine-tune LLMs by learning a prompt prefix, not as tokens, but directly in the latent space, not in the first layer, but in higher layers, with zero-initialized attention (multiply the attention to the added prefix with a gating factor, initialized at zero).

***GPT is becoming a Turing machine:
here are some ways to program it***
A. Jojic et al.

Simulate iterative behaviour (loops) by providing a repetitive execution path as example.

Faithful chain-of-thought reasoning
Q. Lyu et al.

CoT is not faithful: the explanations need not entail the conclusion. Instead, ask the LLM to generate the reasoning and the answer in a formal language (Python, Datalog, PDDL, depending on the task: math, relational inference, robot planning) and verify it.

***Scaling down to scale up:
a guide to parameter-efficient fine-tuning***
V. Lialin et al.

There are many parameter efficient fine-tuning (PEFT) methods:

- *Adapters* are small fully-connected layers (or MoE layers), added (for instance) after transformers;
- *Soft prompts* are prompt prefixes added, not as input tokens, but in latent space (either only for the input layer, or for all layers);
- Selective methods only fine-tune some of the parameters, e.g., only the last layers, or only the biases, or a sparse subset of parameters (subset of important rows and columns, learned mask);
- Reparametrization methods alter the weights W , e.g., $W \leftarrow W + AB'$ (LoRA), $W = W + A \otimes B$ or $W = W + \text{fastfood}(A)$.

Those can be combined.

***Fastfood – approximating kernel expansions
in loglinear time***
Q. Le et al. (2013)

The *fastfood* transform

$$V = \frac{1}{\sigma\sqrt{d}} DH_d G \Pi H_d B$$

where

- $\Pi \in \{0, 1\}^{d \times d}$ is a permutation matrix;
- $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$, $H_{2\ell} = \begin{bmatrix} H_\ell & H_\ell \\ H_\ell & -H_\ell \end{bmatrix}$;
- B is diagonal with ± 1 entries;
- G is diagonal, Gaussian;
- D is diagonal;

approximates Gaussian matrices $Z \sim N(0_{n \times d}, \sigma^2)$.

***Abusing images and sounds for indirect
instruction injection in multi-modal LLMs***
E. Bagdasaryan et al.

With multimodal LLMs, prompt injection is no longer limited to text: you can alter images or sounds to modify the LLM's behaviour, as if you were altering the prompt.

***HuggingGPT: solving AI tasks with ChatGPT
and its friends in Hugging Face***
Y. Shen et al.

Provide HuggingFace models as tools to ChatGPT.

***WizardLM: empowering large language models
to follow complex instructions***
C. Xu et al.

To generate instruction data, start with an initial set of instructions (e.g., “1+1=?”) and ask the LLM to generate new ones, in several directions: adding constraints, deepening, concretizing, increasing the reasoning steps, complicating the input, etc.

***Removing RLHF protections in GPT-4
via fine-tuning***
Q. Zhan et al.

Fine-tune with prompts violating the OpenAI ToS and responses from an uncensored Llama2 70B.

***Measurable Taylor's theorem:
an elementary proof***
G. Viggiano

In the Taylor expansion

$$f(x) = \sum_{j=0}^k \frac{f^{(j)}(c)}{j!} (x-c)^j + \frac{f^{(k+1)}(\xi)}{(k+1)!} (x-c)^{k+1}$$

for $f \in \mathcal{C}^{k+1}$, ξ can be chosen to be a measurable function of x continuous at c .

***Machine learning
for partial differential equations***
S.L. Brunton and J.N. Kutz (2023)

1. To learn a PDE from observational data, PDE-FIND uses

$$u_t = \Theta(u, u_x, u_{xx}, uu_x, \dots) \xi$$

where Θ is a library of candidate terms, e.g.,

divergence	$\nabla \cdot u$
vorticity	$\nabla x u$
shearing	$u_x + v_y$
stretching	$u_x - v_y$

and ξ is a sparse vector.

2. The **Koopman operator** of a discrete, nonlinear finite-dimensional dynamical system

$$x_{t+1} = F(x_t) \quad x_t \in \mathcal{X}$$

is the linear, infinite-dimensional transformation

$$K : \begin{cases} \mathbf{R}^{\mathcal{X}} & \longrightarrow \mathbf{R}^{\mathcal{X}} \\ g & \longmapsto Kg = g \circ F. \end{cases}$$

It satisfies

$$(Kg)(x_t) = g(F(x_t)) = g(x_{t+1}).$$

Learn K satisfying those constraints; diagonalize it,

$$K\phi_k = \lambda_k\phi_k$$

(“intrinsic measurement coordinates”) and reduce the dimension. The non-linear system has become linear – it is easy to solve.

3. If u is a solution of the linear PDE $Lu = f$, then it can be computed from the **Green’s function** of the PDE, *i.e.*, the solution G of $L^\dagger G(\cdot, \xi) = \delta_\xi$, where L^\dagger is the adjoint of L and δ_ξ is the Dirac delta, as follows:

$$\begin{aligned} u(x) &= \langle \delta_x, u \rangle \\ &= \langle L^\dagger G(\cdot, x), u \rangle \\ &= \langle G(\cdot, x), Lu \rangle \\ &= \langle G(\cdot, x), f \rangle \\ &= \int G(\xi, x) f(\xi) d\xi. \end{aligned}$$

The PDE can also be solved by looking for an eigen-decomposition of L :

$$L\phi_n = \lambda_n\phi_n.$$

If a solution u can be represented as a sum of eigenfunctions (Sturm-Liouville theory), then

$$u = \sum \frac{\langle \phi_n, f \rangle}{\lambda_n} \phi_n.$$

Neural operators generalize that to nonlinear PDEs.

Numerically stable parallel computation of (co)variance
R. Schubert and M. Gertz

Do not compute the variance of $\text{Var } X = E[X^2] - E[X]^2$, but (Welford)

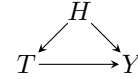
$$\begin{aligned} \mu_{k+1} &= \frac{k}{k+1}\mu_k + \frac{1}{k+1}x_{k+1} \\ S_{k+1} &= S_k + \frac{k}{k+1}(x_{k+1} - \mu_k)^2. \end{aligned}$$

Fundamentals of causal inference with R
B.A. Brumback (2022)

Effect measures include:

$p_0 = E[Y(0)]$	
$p_1 = E[Y(1)]$	
$\frac{p_1}{1-p_1}, \frac{p_0}{1-p_0}$	odds
$RD = p_1 - p_0$	risk difference
$RR = p_1/p_0$	relative risk
$RR^* = \frac{1-p_0}{1-p_1}$	relative risk
$OR = \frac{p_1}{1-p_1} \bigg/ \frac{p_0}{1-p_0}$	odds ratio
$NNT = 1/RD$	number needed to treat
$AF = 1 - 1/RR$	attributable fraction
$CP = \frac{p_1 - p_0}{1 - p_0} = 1 - 1/RR^*$	causal power

Check the R packages: **AER**, **boot**, **car**, **faraway**, **gee**, **geepack**, **Matching**, **resample**.



6. There are several ways of adjusting for a sufficient confounder H .

– Outcome modeling (aka backdoor adjustment), using a model $Y \sim T + H$,

$$E[Y(t)] = \sum_h E[Y|T=t, H=h]P[H=h];$$

– Exposure modeling (aka inverse propensity score weighting), using a model $e : E[T|H] \sim H$,

$$\begin{aligned} E[Y(1)] &= E\left[\frac{TY}{e(H)}\right] \\ E[Y(0)] &= E\left[\frac{(1-T)Y}{1-e(H)}\right] \end{aligned}$$

– Doubly robust standardization

$$\begin{aligned} E[Y(1)] &= E\left[\frac{T}{e(H)}Y - \frac{T-e(H)}{e(H)}\hat{Y}(H, 1)\right] \\ E[Y(0)] &= E\left[\frac{(1-T)}{1-e(H)}Y + \frac{T-e(H)}{1-e(H)}\hat{Y}(H, 0)\right] \end{aligned}$$

– Matching;
– Difference-in-differences (DiD), under the “additive equi-confounding” assumption

$$\begin{aligned} E[Y_1(0)|A=1] - E[Y_1(0)|A=0] &= \\ E[Y_0|A=1] - E[Y_0|A=0] & \end{aligned}$$

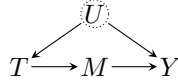
(the difference between the treated and the non-treated, if we had not treated them, is the same as

the difference between those two groups before treatment),

$$\begin{aligned} \text{ATT} &= E[Y_1(1) - Y_1(0)|A=1] \\ &= E[Y_1(1)|A=1] - E[Y_1(0)|A=1] \\ &= E[Y_1|A=1] - \\ &\quad (E[Y_0|A=1] - E[Y_0|A=0] + E[Y_1(0)|A=0]) \\ &= E[Y_1 - Y_0|A=1] - E[Y_1 - Y_0|A=0] \end{aligned}$$

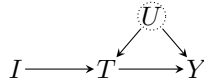
(T is time, A is the treatment).

- Front-door adjustment, given a surrogate marker of the effect of T on Y (a mediator)



$$E[Y(t)] = \sum_m P[M=m|T=t] \sum_{t'} E[Y|M=m, T=t']$$

- Instrumental variables



with *principal stratification*

I : treatment assignment

T : treatment

C : compliance

$T(0) = T(1) = 1$ always taker

$T(0) = T(1) = 0$ never taker

$\forall t A(t) = t$ complier

$\forall t A(t) = 1 - t$ defier

assuming there are no defiers (monotonicity assumption), we can compute the complier average causal effect

$$\begin{aligned} \text{CACE} &= E[Y(1) - Y(0)|C=1] \\ &= \frac{E[Y|T=1] - E[Y|T=0]}{P[A=1|T=1] - P[A=1|T=0]} \end{aligned}$$

- Instrumental variables with *structural nested mean models*: assuming $Y - Y(0) \perp\!\!\!\perp I|T$, we can compute the ATT (with the same formula).

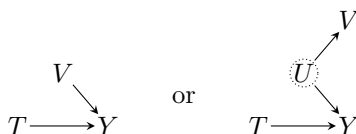
The *propensity score* has many uses:

- Inverse propensity weighting;
- Backdoor adjustment: the propensity score is a sufficient confounder,

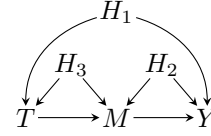
$$\forall t Y(t) \perp\!\!\!\perp T | H \implies \forall t Y(t) \perp\!\!\!\perp T | e(H);$$

- Stratification: computing the ATE in each quartile;
- Matching: easier to do on the propensity score $e(H)$ than on the original confounders H .

Adding *precision variables* reduces variance.



Mediation analysis measures the importance of a mediator.



Denote the potential outcome $T(t, m)$.

$\text{TE} = T(1, M(1)) - Y(0, M(0))$ total effect

$\text{CDE}(m) = Y(1, m) - Y(0, m)$ controlled direct effect

$\text{CIE}(m) = \text{TE} - \text{CDE}(m)$ controlled indirect effect

$\text{NDE} = Y(1, M(0)) - Y(0, M(0))$ natural direct effect

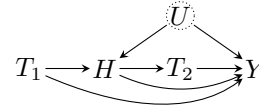
$\text{NIE} = Y(1, M(1)) - Y(1, M(0))$ natural indirect effect

$\text{TE} = \text{CDE} + \text{CIE} = \text{NDE} + \text{NIE}$

$$\text{PE} = \frac{\text{CIE}(m)}{\text{TE}}$$

$$\text{PM} = \frac{\text{NIE}}{\text{TE}}$$

We can also have time-dependent confounders:



we want to know the optimal value of T_1 and, given T_1 and H , the optimal value of T_2 .

Conformal predictive portfolio selection M. Kato (2024)

Consider a model forecasting future returns Y_t from asset features X_t :

$$\hat{Y}_{t+1} = \hat{f}_t(X_{t+1})$$

where \hat{f}_t is trained on $(X_1, Y_1), (X_2, Y_2), \dots, (X_t, Y_t)$. The returns of a portfolio with weights w are $w'Y_t$; the forecasted returns are $w'\hat{Y}_{t+1}$. Apply conformal prediction (CP) to those portfolio return predictions: for each w , we have a prediction interval

$$\hat{C}_t^w(X_{t+1}) = [\underline{r}_{t+1}^w, \bar{r}_{t+1}^w].$$

Find the portfolio weights minimizing some loss function of those intervals

$$w^* = \underset{w}{\text{Argmin}} \ell(\hat{C}_t^w(X_{t+1})).$$

For instance:

- Start with a finite set of portfolios \mathcal{E} ;
- Compute their prediction intervals at level α : $[\underline{r}_w, \bar{r}_w]$;
- Consider a proportion β of portfolios, $\tilde{\mathcal{E}} \subset \mathcal{E}$, with the highest \underline{r}_w (low risk, measured with the VaR);
- Among those, pick the portfolio with the highest \bar{r}_w .

To account for the time series lack of exchangeability, use full conformal prediction, *i.e.*, fit the model on augmented datasets (add (X_{t+1}, y) for all possible values of y) and on all permuted datasets

$$(X_{\sigma(1)}, y_{\sigma(1)}), \dots, (X_{\sigma(t)}, y_{\sigma(t)}), (X_{\sigma(t+1)}, y)$$

for all $\sigma \in \mathfrak{S}_n$. It is actually sufficient to consider only shifts, *i.e.*, permutations of the form

$$\exists j \in \llbracket 0, t-1 \rrbracket \quad \forall i \in \llbracket 1, t \rrbracket \quad \sigma(i) \equiv i + j \pmod{t}$$

**Exact and robust conformal inference methods
for predictive machine learning
with dependent data**

V. Chernozhukov et al.

Conformal prediction for time series: use full conformal prediction and all shifts of the data.

**Applied causal inference
powered by ML and AI**

V. Chernozhukov et al. (2024)

1. To estimate β_1 in $Y = \beta_1 D + \beta_2 W + \varepsilon$, we can use “partialing out”, *i.e.*,

- Compute the residuals \tilde{Y} and \tilde{D} of $Y \sim W$ and $S \sim W$;
- Fit a linear model $\tilde{Y} = \beta_1 \tilde{D} + \eta$.

$$\begin{aligned} \beta_1 &= \underset{b_1}{\operatorname{Argmin}} \operatorname{E}(\tilde{Y} - b_1 \tilde{D})^2 \\ &= \operatorname{E}[\tilde{D}^2]^{-1} \operatorname{E}[\tilde{D} \tilde{Y}] \end{aligned}$$

2. The average treatment effect and the average predictive effect are usually distinct, because of selection bias,

$$\begin{aligned} \text{ATE} &= \operatorname{E}[Y(1) - Y(0)] \\ \text{APE} &= \operatorname{E}[Y|T=1] - \operatorname{E}[Y|T=0] \end{aligned}$$

but, for randomized control trials, they are equal.

Including covariates can help denoise the ATE estimator.

RCTs are not always possible, for instance because of externalities (positive, as for herd immunity after vaccination, or negative, *e.g.*, the drop in college wage premium as more people go to college).

3. The lasso assumes *approximate sparsity*

$$|\beta_j| \leq A j^{-a} \quad \text{for } a > 1/2.$$

Ridge regression assumes the coefficients are dense (small, approximately of the same magnitude). ElasticNet assumes the true model is sparse+dense.

The L^1 penalty is $\lambda \sum |\beta_i|$ only if the predictors are normalized, $\operatorname{E}[X_j^2] = 1$ (if not, multiply λ by $\hat{\sigma}_j$). Choose the penalty scale as

$$\begin{aligned} \lambda &= 2 \cdot c \cdot \hat{\sigma} \sqrt{n} z_{1-a/(2p)} \quad x = 1.1, a = 0.05 \\ z_{1-a/(2p)} &\approx \sqrt{2 \log(2p/a)}. \end{aligned}$$

The lasso sets $\hat{\beta}_j$ to zero if the marginal benefit of moving it away from zero is smaller than the marginal increase in penalty.

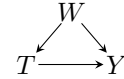
$$\hat{\beta}_j = 0 \quad \text{if} \quad \left| \frac{\partial}{\partial \beta_j} \sum_i (Y_i - \hat{\beta}'_i X_i)^2 \right| < \lambda$$

The **post-lasso** uses the lasso to select the predictors, and then fits an OLS regression (unbiased) on those predictors. Cross-validate the whole post-lasso process, not just the variable selection.

The **lava** method splits the predictors into sparse and dense ones, with L^1 and L^2 penalties respectively.

$$\underset{b=\delta+\xi}{\operatorname{Minimize}} \sum (Y_i - b' X_i)^2 + \lambda_1 \sum \delta_i^2 + \lambda_2 \sum |\xi_i|$$

4. The **double lasso** uses partialing out to infer the effect of T on Y , with high-dimensional confounders W .



- Compute the residuals \tilde{T} , \tilde{Y} of lasso (or post lasso, or any lasso variant) models for $T \sim W$, $Y \sim W$;
- Use a least squares regression $\tilde{Y} \sim \tilde{T}$.

Using the lasso just once (on $Y \sim T + W$, to select the variables) is invalid – the double lasso may select different variables for $Y \sim W$ and $T \sim W$.

5. If we have all the confounders (unconfoundedness, aka conditional ignorability) $\text{CAPE} = \text{CATE}$ and $\operatorname{E}[\text{CAPE}] = \operatorname{E}[\text{CATE}]$.

Conditioning requires you to model $Y|T, X$;

$$\begin{aligned} \delta(x) &= \operatorname{E}[Y(1) - Y(0)|X = x] \\ &= \operatorname{E}[Y|T=1, X=x] - \operatorname{E}[Y|T=0, X=x] \\ \delta &= \operatorname{E}[\delta(X)] \end{aligned}$$

while *propensity score weighting* only requires you to model $T|X$.

$$\begin{aligned} H &= \frac{\mathbf{1}(T=1)}{\operatorname{P}[Y=1|X]} - \frac{\mathbf{1}(T=0)}{\operatorname{P}[Y=0|X]} \\ \delta(X) &= \operatorname{E}[YH|X] \end{aligned}$$

In *stratified RCT*, the propensity score is known.

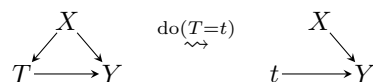
Propensity score weighting gives unbiased estimators, but they may have high variance: also accounting for the link between the confounders X and the outcome Y can “denoise” the estimator.

The ATE can be computed by regressing $Y \sim H$ (clever covariate).

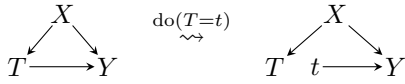
$$\begin{aligned} \phi(t, x) &= \frac{(-1)^t}{\operatorname{P}[T=t|X=x]} \\ H &= \phi(T, X) \\ Y &= \beta H + \varepsilon \\ \text{ATE} &= \operatorname{E}[\beta(\phi(1, X) - \phi(0, X))] \end{aligned}$$

6. Do not condition on colliders.

7. The **do** operation can be represented graphically in two ways, either by removing incoming edges



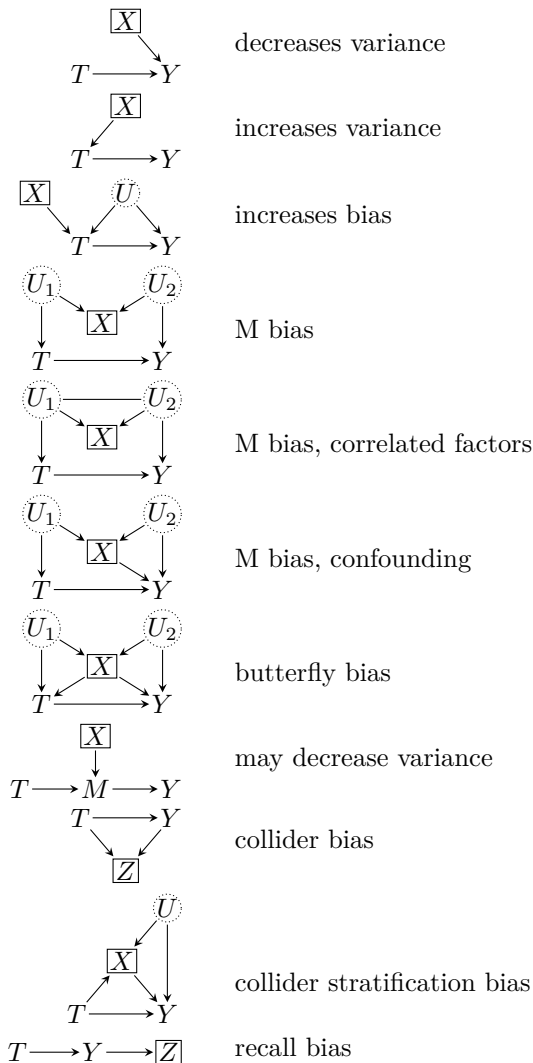
or by splitting the node in two (SWIG, single world intervention graph, of **fix** intervention).



D-separation implies conditional independence, but the converse requires faithfulness. While the model is almost surely faithful, the neighbourhood of unfaithful models can be large, all the more so in high dimensions.

8. There are many *valid adjustment strategies*:

- Parents of the treatment T ;
- Propensity score;
- Parents of the outcome Y (excluding the descendants of T);
- $\text{Parents}(T) \cup \text{Parents}(Y) \setminus \text{Descendants}(T)$ (robust against perturbations of the DAG);
- $\text{Causes}(T) \cap \text{Causes}(Y)$
- Backdoor criterion: a set S blocking all backdoor (*i.e.*, non-causal) paths, and containing no descendant of T ;
- More generally, any set S such that $Y \perp\!\!\!\perp T | S$ in the SWIG.



10. We want to compute

$$\frac{d}{dt} \mathbb{E}[Y|T = t, X = x]$$

or

$$\frac{d}{dt} \mathbb{E}[Y|\text{do}(T = t), X = x]$$

(which reduces to the first expression if X is a sufficient conditioning set).

The partially linear regression model (PLM) is

$$Y = \beta T + g(X) + \varepsilon \quad \mathbb{E}[\varepsilon|T, X] = 0;$$

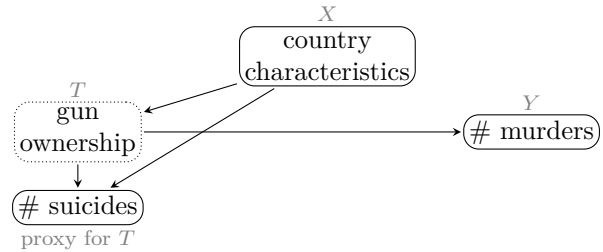
the confounders X have an additive, but nonlinear effect.

If \tilde{Y} and \tilde{T} are the residualized Y and T wrt X , then β is the coefficient of the linear model $\tilde{Y} \sim \tilde{T}$; it is given by the first order condition $\mathbb{E}[(\tilde{Y} - \beta\tilde{T})\tilde{T}] = 0$.

$$\beta = \mathbb{E}[\tilde{T}^2]^{-1} \mathbb{E}[\tilde{T}\tilde{Y}]$$

To prevent overfitting, **Double ML** (DML) uses *cross-fitted residuals*: for observations in fold k , \tilde{Y} and \tilde{T} are estimated from models fitted without fold k

To reduce bias, if you have several models to predict T and Y , use the best ones by looking at the CVMSE.



In the *interactive regression model* (IRM)

$$\begin{array}{lcl} \begin{array}{c} X \\ \swarrow \quad \searrow \\ T \longrightarrow Y \end{array} & \begin{array}{l} T = m(X) + \tilde{T} \\ Y = g(T, X) + \varepsilon \end{array} & \begin{array}{l} \mathbb{E}[\tilde{T}|X] = 0 \\ \mathbb{E}[\varepsilon|T, X] = 0 \end{array} \end{array}$$

the treatment is not additively separable, but it is binary. To estimate the average treatment effect

$$\theta = \mathbb{E}[g(1, X) - g(0, X)].$$

combine regression adjustment

$$\hat{\theta} = \mathbb{E}[g(1, X) - g(0, X)]$$

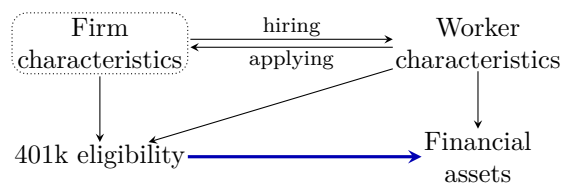
and propensity score reweighting

$$\begin{aligned} \hat{\theta} &= \mathbb{E}[YH] \\ H &= \frac{\mathbf{1}(T=1)}{m(X)} - \frac{\mathbf{1}(T=0)}{1-m(X)} \end{aligned}$$

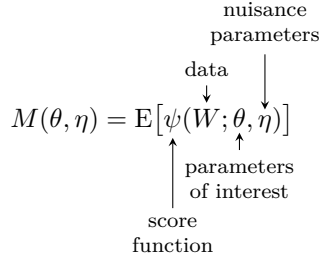
(Horvitz-Thompson transformation): this is the *doubly robust parametrization*

$$\hat{\theta} = \mathbb{E}[g(1, X) - g(0, X) + (Y - g(T, X))H].$$

Propensity scores close to 0 or 1 are problematic: they lead to very large values of H .



More generally, **double machine learning** (DML) uses a *score* function



$$M(\theta, \eta_0) = 0 \iff \theta = \theta_0$$

satisfying the *Neyman orthogonality condition*

$$\left. \frac{\partial M(\theta_0, \eta)}{\partial \eta} \right|_{\eta=\eta_0} = 0$$

i.e., $\hat{\theta}$ is insensitive to small perturbations of η around η_0 : we can use $\hat{\eta}$ instead (from a good machine learning model: lasso, post-lasso, ℓ^1 -penalized neural net, random forest, or ensemble of these). For instance, for the partially linear model (PLM)

$$\begin{aligned} \psi &= (\tilde{Y} - \theta \tilde{T}) \tilde{T} \\ &= [Y - \ell(X) - \theta(T - m(X))](T - m(X)) \\ \eta &= (\ell, m) \text{ nuisance parameters;} \end{aligned}$$

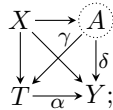
to estimate the ATE in the IRM

$$\begin{aligned} \psi &= [g(1, X) - g(0, X)] + H[Y - g(T, X)] - \theta \\ H &= \frac{T}{m(X)} - \frac{1-T}{1-m(X)} \\ \eta &= (g, m) \end{aligned}$$

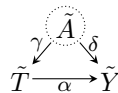
(and similar score functions for the GATE or the ATET).

Use *sample splitting*: estimate $\hat{\eta}_k$ on data without the k th fold, then estimate $\hat{\theta}$ using $\hat{\eta}_k$ for observations in the k th fold; you can also get an estimator of $\text{Var} \hat{\theta}$ and confidence intervals.

12a. We can measure the impact of a non-observed confounder:



first, remove the effect of the observed confounders X

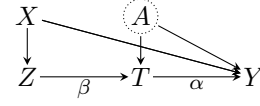


then, project \tilde{Y} on \tilde{T} to get

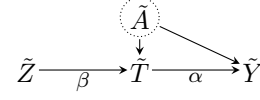
$$\begin{aligned} \beta &= E[\tilde{Y} \tilde{D}] / E[\tilde{D}^2] = \alpha + \phi \\ \phi &= \frac{\delta \gamma}{\gamma^2 + \text{Var}[\varepsilon_T]} = \text{omitted confounder bias.} \end{aligned}$$

Assumptions on the size of δ and γ give bounds on ϕ (alternatively, you can make assumptions on the R^2 of those models).

12b. In presence of an **instrumental variable** (IV)



the effect α of the treatment T on the outcome Y is identifiable



e.g., with the score $\psi = (\tilde{Y} - \alpha \tilde{T}) \tilde{Z}$, leading to

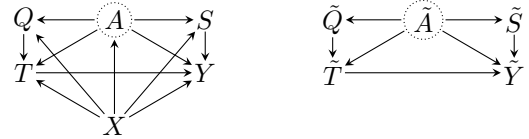
$$\alpha = \frac{E[\tilde{Y} \tilde{Z}]}{E[\tilde{T} \tilde{Z}]}$$

(double machine learning is applicable) or directly, by estimating $\beta\alpha$ and β :

$$\begin{aligned} \beta\alpha &= \frac{E[\tilde{Y} \tilde{Z}]}{E[\tilde{Z}^2]} \\ \beta &= \frac{E[\tilde{T} \tilde{Z}]}{E[\tilde{Z}^2]} \\ \beta\alpha &= \frac{\beta\alpha}{\beta} = \frac{E[\tilde{Y} \tilde{Z}]}{E[\tilde{T} \tilde{Z}]} \end{aligned}$$

(Some of this can be generalized to non-linear IV models.)

12c. *Proxies* of unobserved confounders can also help.



13. With *weak instruments*

$$\begin{aligned} \tilde{Z} &\longrightarrow \tilde{T} \xrightarrow{\alpha} \tilde{Y} \\ \hat{\alpha} &= \frac{E[\tilde{Z} \tilde{Y}]}{E[\tilde{Z} \tilde{T}]}, \quad E[\tilde{Z} \tilde{T}] \approx 0, \end{aligned}$$

use *Neyman's statistic*

$$C(\alpha) = \frac{E[(\tilde{Y} - \alpha \tilde{T}) \tilde{Z}]^2}{\text{Var}[(\tilde{Y} - \alpha \tilde{T}) \tilde{Z}]} \sim N(0, 1)^2 = \chi^2(1)$$

to test for $H_0 : \alpha = \alpha_0$ and derive confidence intervals. This generalizes to other score functions ψ ; in dimension m :

$$\begin{aligned} M(\theta) &= E[\psi(W, \theta, \hat{\eta})] \\ \Omega(\theta) &= \text{Var}[\psi(W, \theta, \hat{\eta})] \\ C(\theta) &= M(\theta)' \Omega(\theta)^{-1} M(\theta) \sim \chi^2(m) \end{aligned}$$

14. “Heterogeneous treatment effects” refers to the CATE, as opposed to the ATE.

$$\begin{aligned} \text{ATE} &= E[Y^{(1)} - Y^{(0)}] \\ \text{CATE} &= E[Y^{(1)} - Y^{(0)} | X = x] \end{aligned}$$

DML works out of the box:

- For models $\hat{\mu}$ and \hat{g} to estimate

$$\mu(Z) = P[T = 1|Z]$$

$$g(Y, Z) = E[Y|T, Z]$$

- Set

$$\hat{Y}_i = H_1(Y_1 - \hat{g}_k(T_i, Z_i)) + \hat{g}_k(1, Z_i) - \hat{g}_k(0, Z_i)$$

$$H_i = \frac{T_1}{\hat{\mu}_k(Z_i)} - \frac{1 - T_i}{1 - \hat{\mu}_k(Z_i)}$$

where $\hat{g}_k, \hat{\mu}_k$ are estimated without the fold containing i ;

- Regress (linearly, if you want confidence intervals) $\hat{Y} \sim X$.

To get valid confidence intervals with random forests, use *honest forests*:

- Use a separate sample to build the tree and to compute the leaf estimates;
- Use smaller subsamples, without replacement, instead of bootstrap samples.

Alternatively, one can look for a *policy* $\pi : X \mapsto \pi(X) \in [0, 1]$ indicating who to treat, maximizing

$$V(\pi) = E[\pi(X)(Y^{(1)} - Y^{(0)})].$$

This does not give a full picture of treatment heterogeneity: if the effects are all negative, $\pi = 0$. Instead, one can look for a quantile-constrained optimal policy

$$\text{Maximize}_{\pi} E[\pi(X)(Y^{(1)} - Y^{(0)})] \quad \text{st} \quad E[\pi(X)] = q.$$

15. Meta learning decomposes the CATE estimation into a sequence of regression problems.

- S (single) learner

$$\hat{g}(T, Z) \approx E[Y|T, Z]$$

$$\hat{\tau}(X) \approx E[\hat{g}(1, Z) - \hat{g}(0, Z)|X]$$

- T (two) learner

$$\hat{g}_1(Z) \approx E[Y|Z, T = 1]$$

$$\hat{g}_0(Z) \approx E[Y|Z, T = 0]$$

$$\hat{\tau}(X) \approx E[\hat{g}_1(Z) - \hat{g}_0(Z)|X]$$

- DR (double robust)

$$\hat{g}_1(Z) \approx E[Y|Z, T = 1]$$

$$\hat{g}_0(Z) \approx E[Y|Z, T = 0]$$

$$\hat{\mu}(Z) \approx E[T|Z]$$

$$\hat{g}(T, Z) = \hat{g}_1(Z)T + \hat{g}_0(Z)(1 - T)$$

$$H = \frac{T}{\hat{\mu}(Z)} - \frac{1 - T}{1 - \hat{\mu}(Z)}$$

$$\hat{Y} = H(Y - \hat{g}(T, Z)) + \hat{g}_1(Z) - \hat{g}_0(Z)$$

$$\hat{\tau}(X) \approx E[\hat{Y}|X]$$

- R-learner

$$\hat{h}(Z) \approx E[Y|Z]$$

$$\hat{\mu}(Z) \approx E[T|Z]$$

$$\check{Y} = Y - \hat{h}(Z)$$

$$\check{T} = T - \hat{\mu}(Z)$$

$$\hat{\tau}(X) = \frac{E[\check{T}^2 \cdot \check{Y}/\check{T}|X]}{E[\check{T}^2|X]}$$

($\hat{\tau}$ is a regression $\check{Y}/\check{T} \sim X$, with weights \check{T}^2);

- X-learner

$$\hat{g}_1(Z) \approx E[Y|Z, T = 1]$$

$$\hat{g}_0(Z) \approx E[Y|Z, T = 0]$$

$$\hat{\mu}(Z) \approx E[T|Z]$$

$$\hat{\delta}_0(Z) \approx E[g_1(Z) - Y|Z, T = 0]$$

$$\hat{\delta}_1(Z) \approx E[g_0(Z) - Y|Z, T = 1]$$

$$\hat{\delta}(Z) = \delta_1(Z)(1 - \hat{\mu}(Z)) + \delta_0(Z)\hat{\mu}(Z)$$

$$\hat{\tau}(X) \approx E[\hat{\delta}(Z)|X].$$

Which model is best depends on the data (imbalance, how complicated \hat{g} and $\hat{\mu}$ are, etc.) – try cross-validation and ensembling.

The optimal (unconstrained) policy only looks at the sign of the CATE

$$\pi^*(X) = \mathbf{1}_{\tau(X) \geq 0}$$

but the misclassification cost is sample-dependent: it is a cost-sensitive classification problem. You may need to penalize the variance of the policy.

16. With longitudinal data, under the conditional parallel trends assumption (parallel trends for $T = 0$ and $T = 1$, once we condition on Z), DML (double/debiased machine learning) can compute the ATET (DiD, difference in differences).

17. If the treatment is assigned (deterministically) by some score $T = \mathbf{1}_{X \geq c}$ (as in regression discontinuity design, RDD), observations close to the threshold are comparable.

$$\tau_{RD} = E[Y(1) - Y(0)|X = c]$$

$$= \lim_{x \rightarrow c^+} E[Y|X = x] - \lim_{x \rightarrow c^-} E[Y|X = x]$$

DAG-GNN: DAG structure learning with graph neural networks
Y. Yu et al. (2019)

One can sample from the linear structural causal model $X = AX + Z, Z \sim N(0, I)$ as

$$X = (I - A)^{-1}Z, \quad Z \sim N(0, I).$$

This can be made nonlinear

$$X = f_2[(I - A)^{-1}f_1(Z)], \quad Z \sim N(0, I)$$

and f_1, f_2 can be learned as an autoencoder

$$\begin{aligned} X &= f_2[(I - A)^{-1}f_1(Z)] && \text{decoder} \\ Z &= f_4[(I - A)f_2(Z)] && \text{encoder.} \end{aligned}$$

The paper uses a VAE.

Gradient-based neural DAG learning

S. Lachapelle et al.

NOTEARS solves the optimization problem

$$\text{Minimize}_{U, V} \text{Mean}_U \|X - XU\|_F^2 + \lambda \|U\|_1 \text{ st } \text{Tr } e^{U \odot U} = d.$$

GraNDAG replaces the linear transformation $X \mapsto XU$ with a neural network, and uses the neural network connectivity matrix

$$C = \left| W^{(L)} \right| \left| W^{(L-1)} \right| \dots \left| W^{(1)} \right|$$

(where $|W|$ is the element-wise absolute value) for the constraint.

CAM: Causal additive models, high-dimensional order search and penalized regression

P. Pühlmann et al. (2014)

CAM is a nonlinear score-based causal discovery algorithm which first looks for a topological order, and only then, the causal graph:

- If there are too many variables (>30), select 10 candidate parents for each node with GAM boosting (preliminary neighbourhood selection);
- Build a complete DAG with unpenalized GES, adding the edge improving the sum of squared residuals (of a GAM) the most;
- Prune the DAG by looking at the p -values of the GAM terms (keep those with $p < 10^{-3}$).

A* Lasso for learning a sparse Bayesian network structure for continuous variables

J. Xiang and S. Kim

GES is a greedy algorithm to find the DAG with the best score. Exhaustive search is unreasonable: there are too many DAGs. But the score is decomposable: the search can be formulated as a shortest path problem in the subset lattice (the value of a subset is the best score attainable with the variables in that subset) which can be solved with dynamic programming. The lasso loss gives a consistent A* heuristic.

A graph autoencoder approach to causal structure learning

I. Ng et al. (2019)

NOTEARS is linear:

$$\text{Minimize}_{A, X_i} \text{Mean}_i \|X_i - AX_i\|^2 + \lambda \|A\|_1 \text{ st } \text{Tr } e^{A \odot A} = d;$$

cGAN generalizes it to nonlinear relations

$$\begin{aligned} \text{Find} & \quad A \\ \text{To minimize} & \quad \text{Mean}_i \|X_i - g_2(Ag_1(X_i))\|^2 + \lambda \|A\|_1 \\ \text{Such that} & \quad \text{Tr } e^{A \odot A} = d \end{aligned}$$

where g_1 and g_2 are pointwise.

Masked gradient-based causal structure learning

I. Ng et al. (2022)

MCSL combines NOTEARS and the Gumbel softmax approach to learn the binary adjacency matrix of a nonlinear structural model

$$X_i = g_i(A_i \odot X) + \varepsilon_1$$

Ordering-based causal discovery with reinforcement learning

X. Wang et al.

Use reinforcement learning to find, not directly the DAG, but just a topological order compatible with it (there are much fewer orders than DAGs):

- State: embedding of the latest variable selected;
- Action: variable to add;
- Reward: BIC improvement;
- Encoder: $\mathbf{R}^{n \times d} \rightarrow \mathbf{R}^{n \times d}$ (self-attention);
- Decoder: LSTM;
- Optimization: policy gradient, actor critic.

Pretrain on simulated data; fine-tune on the data of interest.

$$\text{BIC} = \sum_{jk} \log p(x_{jk} | \text{Pa}(x_{jk}); \theta_j) - \frac{|\theta_j|}{2} \log m$$

Causal discovery with reinforcement learning

S. Zhu et al.

Score-based causal discovery algorithms, such as GES, are greedy – finding the best DAG is NP-hard. Reinforcement learning is an alternative to greedy, exhaustive or random walk search:

- State: solution;
- Action: new (nearby) solution;
- Reward: score improvement.

(You may need to tweak the reward to help the system learn, and you may want to look into variants of reinforcement learning optimizing $\text{Max}_t R_t$ instead of $\sum_t \delta^t R_t$ – *risk-seeking reinforcement learning*.) Here, the approach is slightly different:

- State: bootstrap sample;
- Action: DAG;
- Reward: score + $\lambda_1 \mathbf{1}_{\text{DAG}} + \lambda_2 \text{NOTEARS}$.

Estimation of a structural vector autoregression model using non-Gaussianity
A. Hyvärinen et al.

The VAR model is

$$X_t = \sum_{k=1}^m B_k X_{t-k} + \varepsilon_t.$$

The VAR-LiNGAM model also allows for contemporaneous effects, B_0 , provided they are acyclic and the noise ε_t is non-Gaussian.

$$X_t = \sum_{k=0}^m B_k X_{t-k} + \varepsilon_t$$

The evolving causal structure of equity risk factors
G. D'Acunto et al. (2021)

VAR-LiNGAM, on a moving window, on 11 equity risk factors, with the `lingam` Python package.

Causal inference in R workshop

To estimate causation from observational data with a known causal graph (no unmeasured confounders, and $\forall x P[T = 1|X = x] \neq 0, 1$), one can:

- Adjust for confounders;
- Fit a propensity model $T \sim X$, and use inverse propensity weights (IPW), e.g.,

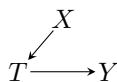
$$w_{\text{ATE}} = \frac{T_i}{p_i} - \frac{1 - T_i}{1 - p_i}$$

(there are other similar formulas for ATT, ATC, ATM, ATO, etc.);

- Use the propensity score to match treated and control samples (`MatchIt`).

These are estimators of the same quantity. Use the bootstrap for the confidence intervals (after reweighting or matching, the observations are no longer independent).

Correlation is sometimes causation (RCT, A/B testing), but even in these cases, you may want to adjust for the confounders or use IPW: the resulting estimators are more efficient.



To graphically assess the propensity scores:

- Histogram of the propensity scores for $T = 0$ and $T = 1$, before and after reweighting;
- “Love plot”, $i \sim \text{smd}_i$, before and after reweighting, where the standardized mean difference (smd) for variable i is

$$\text{smd} = \frac{\bar{x}_{T=1} - \bar{x}_{T=0}}{\sqrt{\frac{s_{T=1}^2 + s_{T=0}^2}{2}}}$$

- ecdf, and weighted acdf.

G-computation (G-formula),

- Fits a model $y \sim T + X$;
- Computes $\widehat{\text{ATE}} = \hat{y}|_{T=1} - \hat{y}|_{T=0}$ (using the model on copies of the data with $T := 0$, resp. $T := 1$, for all the observations)

For continuous exposures, one can:

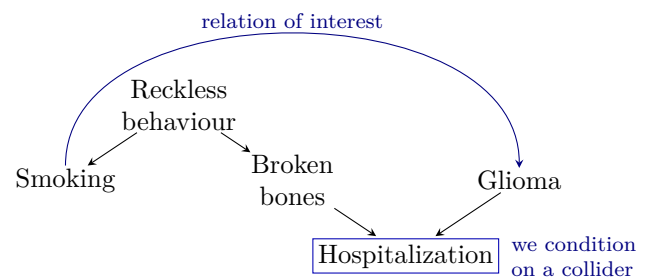
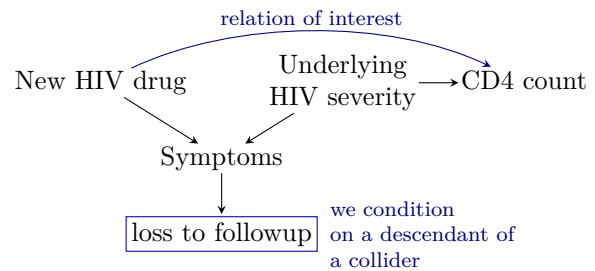
- Fit a linear model $T \sim X$ and convert the residuals into probabilities

$$p = \phi\left(\frac{T - \hat{T}}{\hat{\sigma}}\right)$$

- or bin T into quantiles and use IPW.

Tipping point sensitivity analysis (`tipr`) quantifies unmeasured confoundedness.

To address *selection bias*, fit a probability of censoring model, and use inverse probabilities as weights (multiply the probabilities, if you have several such models).



Causality-inspired models for financial time series forecasting
D.C. Oliveira et al.

Use causal discovery algorithms such as

- SeqICP (invariant causal prediction for sequential data): conditional on its parents X , the distribution of Y does not depend on the environment or period, $Y \perp\!\!\!\perp \text{time} | X$ or

$$Y_t | X_t, t \in I_i \stackrel{d}{=} Y_t | X_t, t \in I_2$$

(but it is overly conservative and tends to return $\text{Pa } Y = \emptyset$);

- Granger causality;
- VAR-LiNGAM;
- Dynotears;
- PCMCi (too slow – their small example would require two weeks of computations).

LiNGAM and Dynotears work better at forecasting monthly S&P 500 returns from macroeconomic variables (FRED-MD, a dataset of 107 macroeconomic variables, with intages, and recommended transformations to make them stationary, updated monthly).

***FRED-MD: a monthly database
for macroeconomic research***
M.W. McCracken and S. Ng (2015)

Bry-Boschan on $\text{cumsum}(\text{PC}_i)$, for $i = 1, 2, 3, 4$, to detect regime changes.

***Enhancing causal discovery in financial
networks with piecewise quantile regression***
C. Cornell et al.

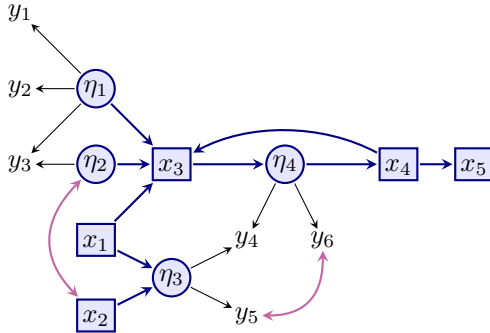
VAR model, with pinball loss, to forecast the 10%, 50%, 90% quantiles, using piecewise linear transformations of the predictors, with knots at the 10% and 90% quantiles to detect tail interactions.

***semopy: a Python package
for structural equation modeling***
G. Mescheryakov and A.A. Igolkina (2019)

A **structural equation model** (SEM) is a model of the form

$$\begin{aligned}\eta &= B\eta + \varepsilon && \text{latent} \\ y &= \Lambda\eta + \delta && \text{observed}\end{aligned}$$

where the sparsity structure of B and Λ (and of $\text{Var } \varepsilon$, $\text{Var } \delta$) is known, e.g.,



It can be estimated by comparing the sample variance matrix of the observed variables, $S = \text{Var } y$, with that of the model, Σ_θ , e.g.,

$$\begin{aligned}\ell(\theta) &= \|\Sigma_\theta - S\|_F^2 \\ \ell(\theta) &= \|I - \Sigma_\theta S^{-1}\|_F^2 \\ \ell(\theta) &= \text{tr}(S\Sigma_\theta^{-1}) + \log \det \Sigma_\theta \quad (\text{Wishart}).\end{aligned}$$

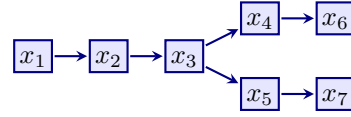
***semopy 2: a structural equation modeling
package with random effects in Python***
G. Moscheryakov et al. (2021)

A **structural equation model** (SEM) is a linear model

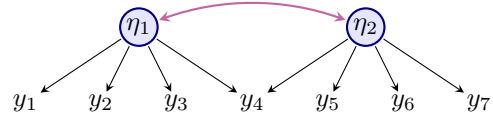
$$x = Ax + \varepsilon$$

where some of the variables are not observed (latent) and the sparsity structure of A and $\text{Var } \varepsilon$ is known. There are special cases:

- A *path analysis* (PA) model only has observed variables



- A *confirmatory factor analysis* (CFA) model has no relations between the latent variables and between observed variables, just relations from latent to observed variables, and covariance between latent variables



Historically, SEMs distinguished between:

- Observed variables (non-leaves: x_i);
- Indicators (leaves: y_i).

Exploratory factor analysis (EFA) looks for a latent structure explaining the observed variables. A heuristic approach could be:

- Compute the correlations $(\rho_{ij})_{1 \leq i, j \leq m}$ and the corresponding distances $D_{ij} = 1 - |\rho_{ij}|$;
- Run the OPTICS clustering algorithm, with `min_cluster_size=2`, to estimate the number k of latent factors;
- Compute a sparse PCA (SPCA) and keep the first k components;
- Refine the CFA by fitting the SEM and dropping the coefficients with a large p -value.

Random forests for change point detection
M. Lonschien et al. (2023)

To find a breakpoint, compare the performance of classification models predicting, from x_t , if $t \in [t_0, \tau]$ or $t \in [\tau, t_1]$: the best performance is obtained if τ is a breakpoint. This can be generalized to several (an unknown number of) breakpoints, and efficiently approximated.

Alternatives include:

- E-divisive (ECP), which finds breakpoints minimizing the energy distance between segments; the *energy distance* between two random variables can be computed from their characteristic functions:

$$\begin{aligned}\mathcal{E}_a(X, Y) &= \int_{\mathbf{R}^d} |\phi_X(t) - \phi_Y(t)| w(t) dt \\ w(t) &\propto |t|^{-d-\alpha} \\ \alpha &= 2.\end{aligned}$$

***A kernel multiple change-point algorithm
via model selection***
S. Arlot et al. (2012)

Kernel change point methods (KCP) minimize (with dynamic programming) the kernel least squares criterion

$$R(\tau) = \frac{1}{n} \sum_{i=1}^n k(x_i, x_i) - \frac{1}{n} \sum_{\ell} \frac{1}{\tau_{\ell} - \tau_{\ell-1}} \sum_{i,j \in [\tau_{\ell-1}+1, \tau_{\ell}]} k(x_i, x_j)$$

with a penalty for the number D of segments

$$\frac{1}{n} \left[c_1 \log \binom{n-1}{D-1} + c_2 D \right].$$

The loss can also be written

$$R(\tau) = \frac{1}{n} \|\Phi(\mathbf{x}) - \Pi_{\tau} \Phi(\mathbf{x})\|^2$$

where

$$\begin{aligned} k(x, y) &= \langle \Phi(x), \Phi(y) \rangle \\ \Phi : \begin{cases} \mathcal{X} & \longrightarrow \mathcal{H} \\ x & \longmapsto k(x, \cdot) \end{cases} & \text{RKHS} \\ \mathbf{x} &= (x_1, \dots, x_n) \\ \Phi(\mathbf{x}) &= (\Phi(x_1), \dots, \Phi(x_n)) \end{aligned}$$

F_{τ} is the subset of \mathcal{H}^n of locally constant sequences with locally constant pattern (coarser than) τ ; Π_{τ} is the orthogonal projection in F_{τ} .

For the d -dimensional simplex (e.g., if x_i is a histogram – histogram features are often used for image or audio data), use the χ^2 kernel

$$k(x, y) = \exp - \frac{1}{h \cdot d} \sum_{i=1}^d \frac{(x_i - x_j)^2}{x_i + x_j}$$

(for some bandwidth $h > 0$).

Recommendation with generative models
Y. Deldjoo et al. (2024)

Classical RecSys data contains user, item, timestamp, feedback (explicit (rating) or implicit (click)).

Matrix factorization models the feedback as

$$R_{ui} = \gamma_u \cdot \gamma_i;$$

one can add offset and bias

$$R_{ui} = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i$$

or include the previous item j

$$R_{uij} = \gamma_u \gamma_i + \beta_i^{\text{next}} \beta_j^{\text{previous}}.$$

User-free models only look at the similarity with items in the user's history

$$R_{ui} = \text{Mean}_{j \in u \setminus \{i\}} \gamma_i \cdot \gamma_j^{\text{history}}.$$

Generative models produce a *probabilistic forecast*: there is not a single output, but a probability distribution of outputs, from which we can sample.

A user's history is a sequence of tokens: we can use LLM-like models to forecast the next token.

RecSys can generate sets (or sequences) of items instead of isolated items: the items should be complementary or substitutable (ex: outfits).

GANs can improve traditional RecSys training by providing hard negatives (instead of uniformly sampled ones). But GANs can also be used to model normal users, and create hard-to-detect fake profiles.

Generative models also provide data augmentation.

Traditional RecSys, when followed by LLM-based reranking, can be seen as the retrieval part of a RAG system.

The data can also contain text (user profiles, item reviews) and metadata.

Conversational RecSys provide textual explanations (personalized reviews), and allow for interactive query refinement – but also personalized content and advertising.

***A contextual bandit approach to personalized
news article recommendation***
L. Li et al. (2010)

The UCB algorithm, for multi-arm bandits (MAB), chooses the arm i maximizing

$$\text{UCB}_i = \hat{\mu}_i + \sqrt{\frac{2 \log t}{n_i}}.$$

For contextual bandits, use

$$\begin{aligned} \text{LinUCB}_i &= x_i' \hat{\theta}_i + \alpha \sqrt{x_i' A_i^{-1} x_i} \\ A_i &= X_i' X_i + I \\ \hat{\theta}_i &= (X_i' X_i + I)^{-1} X_i' b_i \quad (\text{ridge}) \\ x_i &: \text{feature vector} \\ X_i &: \text{design matrix} \\ b_i &: \text{response vector} \end{aligned}$$

This can be generalized to hybrid models (some parameters are shared among the arms).

***Thompson sampling
for contextual bandits with linear payoffs***
S. Agrawal and N. Goyal (2013)

Thompson sampling can be generalized to contextual MABs:

- Compute the posterior distribution of θ_i , for each i : $N(\hat{\theta}_i, \alpha^2 A^{-1})$;
- Sample from them;
- Choose the arm i maximizing $x_i' \theta_i$.

LinTS is similar to LinUCB: the former picks the arm maximizing a random sample, the latter an arm maximizing an upper bound.

Value-difference based exploration: adaptive control between epsilon-greedy and softmax
M. Tokic and G. Palm

The *softmax* MAB algorithm is a modification of ϵ -greedy paying attention to the difference in rewards: it picks the arm i with probability $p_i \propto \exp(\hat{\mu}_i/\tau)$.

Algorithms for the multi-armed bandit problem
V. Kuleshov and D. Precup (2009)

The *pursuit* MAB algorithm is

$$p(t+1) = p(t) + \beta[e_i - p(t)]$$

where e_i is the basis vector for

$$i = \underset{i}{\operatorname{Argmax}} \hat{\mu}_i(t).$$

The *Boltzman* (softmax) algorithm uses

$$p_i(t+1) \propto \exp \hat{\mu}_i(t)/\tau.$$

The *reinforcement comparison* method uses “preferences” π instead of average rewards $\hat{\mu}$:

$$p_i(t+1) \propto \exp \pi_i(t)$$

where $\pi_i(t+1) = \pi_i(t) + \beta[r(t) - \bar{r}(t)]$ if arm i is chosen at time t and gives reward $r(t)$.

There are many variants of *UCB*:

$$j(t) = \underset{i}{\operatorname{Argmax}} \hat{\mu}_i + \sqrt{\frac{2 \log t}{n_i}}$$

$$j(t) = \underset{i}{\operatorname{Argmax}} \hat{\mu}_i + \sqrt{\frac{\log t}{n_i} \operatorname{Min}\left\{\frac{1}{4}, V_i\right\}}$$

$$V_i(t) = \hat{\sigma}_i(t) + \sqrt{\frac{2 \log t}{n_i(t)}}$$

Introduction to conformal prediction with Python
C. Molnar (2023)

6. Conformal prediction (CP) works as follows:

- Split the data into training and calibration (and test, and validation) sets;
- Train the model on the training set
- Compute a *non-conformity score* for all observations in the calibration set (any measure of how unusual y is, given x), e.g., $1 - \hat{p}(y|x)$, for a classification model;
- Compute its $1 - \alpha$ quantile \hat{q} (with a finite-sample adjustment);
- Given new data, x , return all the y ’s whose non-conformity score is below \hat{q} as *prediction set*.

(This is inductive CP; transductive CP uses the whole dataset, but needs to refit the model many times). For the splits, you can use a single split, cross-validation (CV) or leave-out-one (LOO).

7. For classification:

- The *score method* uses the probabilities of the true classes to compute the conformity scores: $s_i = 1 - \hat{p}(x_i)_{y_i}$ (where $\hat{p}(x_i)$ is the vector of predicted probabilities); it is not *adaptive*: it only guarantees *marginal coverage*, not *class-conditional coverage* – the coverage is $1 - \alpha$ only on average, but it can be much higher/lower for easier/harder classes.
- *Adaptive prediction sets* (APS) use cumulated probabilities, from the largest down to the true class; at test time, compute the cumulated sorted probabilities, and return the classes until the threshold (you can include the label just above the threshold, or not (but this can lead to empty prediction sets), or (better) decide at random);
- Top- k uses the rank of the true class; the prediction sets all have the same size;
- Regularized APS (RAPS) penalizes the inclusion of too many classes;
- Group-balanced conformal prediction applies conformal prediction separately for each group;
- Class-conditional APS uses group-balanced CP for the classes to predict: since they are not known at test time, it computes the CP for all classes, and returns the union of the forecasts.

8. For regression:

- $s(y, x) = |y - \hat{f}(x)|$ gives prediction intervals all of the same size (this is fine if the data is homoskedastic);
- Locally adaptive CP

$$s(y, x) = \frac{|y - \hat{f}(x)|}{\hat{\sigma}(x)}$$

accounts for heteroskedasticity (but can output excessively large intervals if the data is homoskedastic).

Instead of conformalizing (mean) regression, we can conformalize quantile regression

$$s(y, x) = \operatorname{Max}\{\hat{f}_{\text{low}}(x) - y, y - \hat{f}_{\text{up}}(x)\}$$

(the distance to the nearest interval boundary, with a negative sign if inside).

9. Conformal prediction can be generalized to time series (EnbPI, in spite of non-exchangeability), multi-label classification, outlier detection (CP provides accurate control of the false positive rate).

Implementation: `mapie`; also check `crepes`, `nixtla`.

Recipe for a general, powerful, scalable graph transformer
L. Rampásek et al.

GraphGPS combines (adds) MPNN layers (GIN with edge features, gated GCN, GRU in the depth direction, PNA) and global attention (transformer, performer, big bird). Input features are preprocessed by an MLP or some variable-input-size network (DeepNet, SignNet).

Design space for graph neural networks
J. You et al.

GraphGym is (was?) a descriptive Python library for GNNs, built on top of PyG, to explore the design space of GNNs:

- BatchNorm (yes/no);
- Dropout (0, 0.3, 0.6);
- Activation (ReLU, PReLU, Swish);
- Aggregation (mean, max, sum);
- Connectivity (stack, skip-sum, skip-cat);
- Pre-processing (number of MLP layers);
- Post-processing;
- Message passing (number of layers);
- Batch size;
- Learning rate;
- Optimizer (SGD, Adam);
- Training epochs.

The curse of recursion: training on generated data makes models forget
O. Shumailov et al.

Training on generated data causes model collapse:

- At each generation, since the data is finite, there is a nonzero probability that some of the data distribution will be lost; the tails will disappear, and the distribution will progressively drift;
- Since the model is only an approximation (it does not perfectly reproduce the target distribution), the generated data will follow a slightly different distribution, for instance, assigning a nonzero probability outside the support of the target distribution.

Differentiable Euler characteristic transforms for shape classification
E. Röell and B. Rieck

The Euler characteristic of a simplicial complex K is

$$\chi(K) = \sum_{n \geq 0} (-1)^n |K^n|$$

where $|K^n|$ is the number of n -simplices. If the 0-simplices $\sigma \in K^0$ have coordinates $x_\sigma \in \mathbf{R}^n$, a function $f : \mathbf{R}^n \rightarrow \mathbf{R}$ defines a function $\tilde{f} : K \rightarrow \mathbf{R}$ by

$$\begin{aligned} \tilde{f}(\sigma) &= x_\sigma & \text{for 0-simplices} \\ \tilde{f}(\sigma) &= \max_{\tau \subset \sigma} \tilde{f}(\tau) & \text{for higher-dimensional simplices} \end{aligned}$$

and a filtration $K_r = \tilde{f}^{-1}((-\infty, r])$.

The *Euler characteristic transform* (ECT) of K is the function

$$\begin{cases} \mathbf{S}^{n-1} \times \mathbf{R} & \longrightarrow \mathbf{Z} \\ (\xi, h) & \longmapsto \chi(\tilde{f}_\xi^{-1}((-\infty, h])) \end{cases}$$

where $f_\xi = \langle \cdot, \xi \rangle$.

It can be written as a sum of step functions

$$\text{ECT}(\xi, h) = \sum_k (-1)^k \sum_{\sigma_k} \mathbf{1}_{[\tilde{f}_\xi(\sigma_k), \infty)}(h).$$

To make it differentiable, replace the step functions with sigmoids.

Beyond word frequency: bursts, lulls, and scaling in the empirical distribution of words
E. G. Altmann et al.

Words have different recurrence time distributions; they can be modeled with a “stretched exponential” (aka Weibul)

$$F_\beta(\tau) = \exp(-a\tau^\beta).$$

This can be explained with a renewal process: the probability of using a word decays as a power law since the last use of that word.

Retrieve, merge, predict: augmenting tables with data lakes (experiment, analysis and benchmark paper)
R. Cappuzzo et al.

Given a table and a data lake (a large set of database tables), identify joinable tables, and join them, greedily, for best performance in some downstream task.

Neural factors: a novel factor learning approach to generative modeling of equities
A. Gopal

An autoencoder-based factor model can generate synthetic data:

- A neural network converts stock characteristics (financial statements, a few reference indices) into factor exposures β_{ijt} and the parameters of a Student distribution $\alpha_{it}, \sigma_{it}, \nu_{it}$;
- The factors follow a Gaussian distribution $z_{t+1} \sim N(0, I)$;
- The stock returns are then

$$r_{i,t+1} \sim T_{\nu_{it}}(\alpha_{it} + \beta_{i,t}^\top z_{t+1}, \sigma_{it}).$$

A new measure of risk using Fourier analysis
M. Grabinski and G. Klinkova (2024)

Use the Fourier transform (on detrended log-prices) to decompose price changes into contributions of different frequencies: if they mostly come from high frequencies, this is speculation.

Modeling arterial and venous flows in Japanese supply chain: a network-based approach to the circular economy
H. Goto

The supply chain has a bow-tie structure (the largest strongly connected component, its upstream, its downstream, the rest of the giant weakly connected component, and the rest). Take an industry classification, assume it is ordered (sic), and label flows as “arterial” or “venous”, depending on their direction. [Instead, I would take the order maximizing arterial flow.]

**Analysis of cross-shareholding network
Japanese listed companies
S. Tanabe and T. Ohnishi**

Evolution of the bow-tie structure.

**Dynamical analysis of financial stock network:
improving forecasting using network properties
I. Aчитouv**

Forecast hourly or daily returns of S&P 500 stocks from node or graph features:

- Centralities (degree, closeness, betweenness, eigenvalue);
- Clustering coefficient;
- Modularity;
- Largest component;
- Resilience (size of the largest component after removing a fraction of the nodes).

Build the graph by thresholding the correlation matrix; choose the smallest threshold that makes the degree follow a power law distribution (increase the threshold until the degree distribution becomes convex: $\rho = 0.9$).

**Regime-aware factor allocation
with optimal feature selection
T. Bosancic et al.**

The discrete jump model is

$$\text{Minimize}_{\theta_1, \dots, \theta_K \in \mathbf{R}^D, s \in \llbracket 1, K \rrbracket^T} \sum_t \ell(y_t, \theta_{s_t}) + \lambda \sum \mathbf{1}_{s_{t-1} = s_t}.$$

The continuous jump model replaces states $s \in \llbracket 1, K \rrbracket$ with probabilities $s \in \Delta_K$.

$$\text{Minimize}_{\theta_1, \dots, \theta_K \in \mathbf{R}^D, s_1, \dots, s_T \in \Delta_K} \sum_t L(y_t, \Theta, s_t) + \frac{\lambda}{4} \sum_t \|s_{t-1} - s_t\|_1^2$$

where $L(y_t, \Theta, s_t) = \sum_k s_{tk} \ell(y_t, \theta_k)$

**Graph learning
(ICML 2024 Tutorial)**

Early graph learning methods relied on node embeddings: DeepWalk, Node2Vec (a biased random walk, between depth-first search and breadth-first search). The first graph neural nets relied on message passing (MPNN), which can be written with matrix multiplications (GCN), and augmented with a gating mechanism (GAT, graph attention); if there are high-degree nodes, use sampling (GraphSAGE samples the 2-hop neighbourhoods)

Spectral graph theory studies the graph Laplacian. The first eigenvectors (“eigenfunctions”) minimize the variability of the signal. The Cheeger constant is the minimum (normalized) cut (NP hard)

$$h_G = \min_{S \subset V} \frac{|\partial S|}{\min\{\text{vol}(S), \text{vol}(\bar{S})\}}$$

(the volume is the sum of the degrees); it is related to the second eigenvalue

$$\frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}.$$

The commute time CT (of a node u) is the expected number of steps needed for a random walk starting at u to come back to u . The effective resistance is

$$R_{uv} = \frac{\text{CT}_{uv}}{\text{vol}(G)}.$$

A *graph transformer* is a transformer (not a GNN) whose positional encoding comes from a graph:

- Laplacian eigenvectors (which generalize sines and cosines);
- Learned positional embeddings (LPE), *i.e.*, learned transformations of

$$\begin{pmatrix} \lambda_1 & \phi_{1j} \\ \vdots & \vdots \\ \lambda_m & \phi_{mj} \end{pmatrix}$$

- Centrality encoding $h_i = x_i + z_{\deg^-(v_i)}^- + z_{\deg^+(v_i)}^+$;
- Spatial encoding.

The attention matrix can be sparse: expormer uses the original graph, an expander graph, and a global sink (not unlike BigBird). GraphGPS combines MPNN layers, attention layers, and positional/structural encodings.

The expressivity of GNNs is limited by the WL isomorphism test – but 1-WL cannot distinguish d -regular graphs... To enhance expressivity, one can:

- Add features: random features, substructures, affinity measures (effective resistance, hitting time, etc.);
- Modulate message passing (e.g., with gating, in GAT);
- Modify the graph, e.g., higher-order GNNs for message passing on k -tuples of nodes instead of nodes.

The Dirichlet energy measures *oversmoothing*

$$E(H) = \text{tr}(H' L H) = \frac{1}{2} \sum_{(u,v) \in E} \left\| \frac{h_u}{\sqrt{d_u}} - \frac{h_v}{\sqrt{d_v}} \right\|^2$$

(the Rayleigh coefficient, used to define the eigenvalues, is a normalized Dirichlet energy). The rate of convergence of a random walk $x_t = P^t x_0$, resp. the heat kernel, $x_t = e^{-tL}$ to the stationary (resp. uniform) distribution is λ_2 : convergence is faster if the total effective resistance is low. To limit oversmoothing:

- Normalize the node embeddings;
- Sparsify the graph (this reduces the spectral gap);
- Regularize the weight matrix;
- Add skip connections;
- Change the GNN dynamics (GraphSAGE, GAT, PIGNN, AdaptiveGNN).

Oversquashing (the exponential growth of the perceptive field) can be measured by the Cheeger constant, the effective resistance, the curvature, the Hessian.

$$\text{sensitivity}(u \rightarrow v) = \left\| \frac{\partial h_v^{(r)}}{\partial h_u^{(0)}} \right\|$$

The symmetric Jacobian obstruction

$$\left(\frac{1}{d_v} \frac{\partial h_v^{(m)}}{\partial h_v^{(k)}} - \frac{1}{\sqrt{d_u d_v}} \frac{\partial h_v^{(m)}}{\partial h_u^{(k)}} \right) + \left(\frac{1}{d_u} \frac{\partial h_u^{(m)}}{\partial h_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}} \frac{\partial h_u^{(m)}}{\partial h_v^{(k)}} \right)$$

is related to the (total) effective resistance.

Solutions include:

- Graph rewiring: spacial (SDRF, curvature) or spectral (DiffWire);
- Virtual nodes;
- Advanced architectures: AdaptiveGNN.

There is a trade-off between oversmoothing (OSM) and oversquashing (OSQ): high R , low λ_2 gives more OSQ, low R , high λ_2 gives more OSM.

Data-efficient machine learning (ICML 2024 Tutorial)

Data-efficient ML looks for a small subset of the training data, such that a model trained on that subset has a generalization error similar to that of a model trained on the whole data.

$$\text{Maximize } F(S) \text{ st } |S| \leq k \\ \text{SCV}$$

1. For supervised learning, look for a subset with similar gradients. For a single point $w \in \mathcal{W}$, take the medoids of $\{\nabla f_i(w), i \in V\}$, weighted with the cluster sizes. It is a submodular problem (diminishing returns): it can be solved (approximately) with a greedy algorithm

$$F(D^*) = \sum_{i \in V} \text{Min}_{j \in S^*} \|\nabla f_i(w) - \nabla f_j(w)\|.$$

This depends on w , but we can take the worst case:

$$F(S^*) \leq \sum_{i \in V} \text{Min}_{j \in S^*} \underbrace{\text{Max}_{w \in \mathcal{W}} \|\nabla f_i(w) - \nabla f_j(w)\|}_{d_{ij}}.$$

If f is convex,

$$d_{ij} \leq \text{const} \cdot \|x_i - x_j\|$$

where x are the feature vectors: it suffices to cluster the features. If f is not convex (e.g., a neural net), we still expect that

$$d_{ij} \leq \text{const} \cdot \left\| \nabla_{z_i^{(L)}} f_i(w) - \nabla_{z_j^{(L)}} f_j(w) \right\|.$$

(gradients wrt the last layer): for classification, we would cluster the log-probabilities; since they depend on w , we need to update the clusters from time to time.

To decide when to update the coresets, model the loss as a piecewise convex function, and change the coreset when the model moves to a different convex region: check if the current convex approximation remains valid.

$$\tilde{\ell}(\delta) = \ell(w) + g_s \delta + \frac{1}{2} \delta^\top H_s \delta \\ \frac{|\tilde{\ell}(\delta) - \ell(w + \delta)|}{\ell(w + \delta)} \leq \tau$$

To ensure SGD gets unbiased gradient estimates, select multiple random subsets of size $r \gg 1$, and form a coreset (minibatch) of size m for each of them; check the convex approximation on the union of the minibatches. As training progresses, this selects increasingly difficult examples.

There are simple heuristics, looking for difficult-to-learn examples, but they are affected by noisy labels:

- Drop unforgettable samples (a “forgetting event” occurs when a sample, which was correctly classified earlier in training, becomes incorrectly classified);
- Train several models, for only 20 epochs, and keep samples with the largest dLoss/dLastLayer;
- Cartography the dataset into:
 - High confidence, low variability: easy – discard;
 - Low confidence, low variability: potentially mislabeled – discard;
 - High variability (ambiguous), *i.e.*, the true class probability fluctuates during training – keep.

Those heuristics allow the removal of up to 30% of the data; if you want to remove more, keep the easy examples instead of the hard ones. Those heuristics do not take similarity into account; submodular optimization does, and yields a curriculum: it starts with easy examples (but not the easiest), and progressively increases the difficulty (measured by “forgettability”).

2. For self-supervised contrastive learning, this is trickier: the loss

$$\ell_{ij} = -\log \frac{\exp \text{sim}(z_i, z_j) / \tau}{\sum_{k \neq i} \exp \text{sim}(z_i, z_k) / \tau}$$

(and its gradient) for an observation depends on all the observations – it tries to

- Align augmentations of the same example (“alignment”);
- Push apart augmentations of different examples (or examples from different classes) (“divergence”).

The coreset should preserve alignment and divergence.

Define the expected alignment distance as

$$d_{ij} = \mathbb{E}_{\substack{x \in A(x_i) \\ x' \in A(x_j)}} \|x - x'\|$$

from a cheap proxy model.

To preserve alignment, ensure that, for each discarded example, there is another with similar augmentations

$$\forall u \in V_k \setminus S_k \quad \min_{j \in S_k} d_{ij} \leq \delta,$$

i.e., we select a diverse set of examples in each latent class.

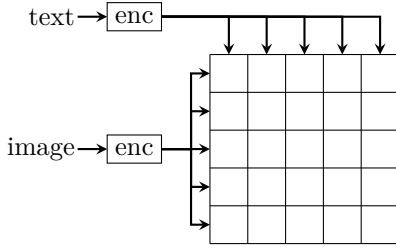
The latent classes are, for instance, k -means clusters on a cheap proxy model, or a foundational model (CLIP).

To preserve divergence, find a subset preserving the centers of the latent classes (*i.e.*, select the most central examples in each latent class)

$$\underset{\substack{S \subset V \\ |S| \leq r}}{\text{Minimize}} \quad \sum_{i \in V_k \setminus S_k} \sum_{j \in S_k} d_{ij}.$$

(This is a submodular optimization problem.) Contrary to supervised learning, where we were discarding easy examples and keeping those near decision boundaries, for SLL, we train on easy examples (cluster centers).

3a. Multimodal foundational models (CLIP, contrastive language-image pretraining)



are similar: they pull together matching image-caption pairs, and push apart other pairs, but they are even more data-hungry (400 times what supervised learning requires, versus 10 times).

For diversity (alignment of modalities), find a subset preserving the cross-covariance in each latent class

$$C = \frac{1}{|V|} \sum_{i \in V} (x_{\text{vision}}^i - \mu_{\text{vision}})(x_{\text{language}}^i - \mu_{\text{language}}).$$

For centrality, preserve the latent class centers. Define the latent classes with the cross-modality similarity (from a proxy model)

$$\text{sim}(\text{caption}_1, \text{image}_2) + \text{sim}(\text{caption}_2, \text{image}_1).$$

The corresponding coresets can be

$$\begin{aligned} & \underset{S_k \subset V_k}{\text{Maximize}} \quad \sum_{i \in S_k} \text{csim}(i, i) \\ & \underset{S_k \subset V_k}{\text{Maximize}} \quad \sum_{j \in S_k} \sum_{i \in V_k} \text{csim}(i, j) \\ & \underset{S_k \subset V_k}{\text{Maximize}} \quad \sum_{\substack{i \in V_k \\ j \in S_k}} \text{csim}(i, j) + \sum_{i \in S_k} \text{csim}(i, i) \end{aligned}$$

(these are non-monotone submodular optimization problems).

Heuristics include:

- Deduplication: cluster the CLIP embeddings, discard samples when the similarity is above $1 - \varepsilon$;
- Focus on examples that are learnable but not easy to learn

$$\text{CLIP}_{\text{fully trained}}(i, i) - \text{CLIP}_{\text{partially trained}}(i, i)$$

(on a clean dataset, those heuristics do not improve on a random subset).

3b. For LLMs, we cannot match the gradients: they are too high-dimensional, even for the last layer, even for LoRA (for fine-tuning). If the curvature is small (during fine-tuning, it is), examples with similar loss trajectories (similar values for a few points – with a smaller model) have similar gradients: can sample from loss-trajectory clusters.

If you want to actually match the gradients:

- Use the V projection (from the transformer's Q , K , V matrices) at the last layer;
- Use zeroth order gradients (gradients as an expectation)
- Sparsify (keep 1000 or 2000 dimensions with the largest magnitude).

Heuristics include:

- Perplexity (prefer examples with average perplexity);
- Memorization ranking;
- Deduplication;
- Data selection with another LLM;
- Centroids of hidden states;
- Influence functions (if there is a target class);
- Data models.

Neural operator learning (ICML 2024 tutorial)

The first layer of a neural net can be written

$$b_i = \sigma \left(\frac{1}{n} \sum_j k_{ij} a_j \right)$$

$$\text{or} \quad v(y_i) = \sigma \left(\sum \kappa(y_i, x_i) a(x_j) \Delta x_j \right)$$

$$\text{i.e.,} \quad v(y) = \sigma \left(\int \kappa(y, x) a(x) dx \right);$$

it is a linear integral operator. We can add bias and skip connection

$$v(y) = \sigma \left(\int \kappa(y, x) a(x) dx + W a(y) + b(y) \right).$$

The output function can be evaluated at any point, and the input can be provided at any discretization.

The graph neural operator (GNO) implements κ as a neural net.

The Fourier neural operator uses basis functions

$$a \rightarrow \mathcal{F} \rightarrow \kappa \rightarrow \mathcal{F}^{-1} \rightarrow v$$

(on a regular grid, use the FFT).

The model can be trained on a coarse grid and evaluated on a finer one.

The integral can also be computed with:

- Gaussian quadrature;
- The multipole method (UNO: U-shaped NO, similar to UNet).

This also works for other network architectures:

- Transformers (just replace \sum with \int);
- CNNs (e.g., derivatives).

Physics-informed neural operators (PINO) use both data and a PDE (the PDE can have a higher resolution).

Mixtures of experts in the era of LLMs (ICML 2024 tutorial)

(Sparse-gated) mixtures of experts (MoE) can suffer from:

- Imbalanced routing (some experts are used too often, or too rarely);
- Redundant experts.

Many design choices are possible:

- Top-2 or fine-grained;
- Shared expert (an expert that is always used);
- Experts after each layer, with more experts in the last layers;
- Loss function.

To convert a dense LLM into a MoE, you can:

- Copy the feed-forward part to form experts (this increases the number of parameters);
- Split the feed-forward part (using k -means on the neuron activations).

Distribution-free predictive uncertainty quantification: strength and limits of conformal prediction (ICML 2024 tutorial)

Given observations $(X_1, Y_1), \dots, (X_n, Y_n)$, conformal prediction aims to find C_α such that

$$\mathbb{P}[Y_{n+1} \in C_\alpha(X_{n+1})] \geq 1 - \alpha.$$

Split conformal prediction (SCP) proceeds as follows:

- Split the data into training (75% to 90%), calibration and test (X_{n+1}) ;
- Fit a model $\hat{\mu}$ on the training set;
- Compute “conformity scores” on the calibration set

$$S_{\text{Cal}} = \{|\text{residuals}| \text{ on Cal}\} \cup \{\infty\}$$

- Use the corresponding quantiles

$$\hat{C}_\alpha(X_{n+1}) = [\hat{\mu}(X_{n+1}) \pm q_{1-\alpha}(S_{\text{Cal}})].$$

If the data is exchangeable, we have marginal validity

$$1 - \alpha \leq \mathbb{E}_{X_{n+1}} \mathbb{P}[Y_{n+1} \in \hat{C}_\alpha(X_{n+1})] \leq 1 - \alpha + \frac{1}{\#\text{Cal} + 1}$$

but SCP is not adaptive (it does not recognize heteroskedasticity).

Conformalized quantile regression (CQR) fits two quantile models, e.g., 10% and 90%, and uses the signed distance to the closest quantile (negative if inside, positive if outside).

$$\hat{C}_\alpha(X_{n+1}) = [\widehat{\text{QR}}_\ell(X_{n+1}) - q_{1-\alpha}(S_{\text{Cal}}), \widehat{\text{QR}}_u(X_{n+1}) + q_{1-\alpha}(S_{\text{Cal}})]$$

More generally, if $S = \{s(x_i, y_i), i \in \text{Cal}\} \cup \{\infty\}$,

$$\hat{C}_\alpha(X_{n+1}) = \{y : s(X_{n+1}, y) \leq_{1-\alpha}(S)\}.$$

Locally weighted SCP uses

$$s(X, Y) = \frac{|\hat{\mu}(X) - Y|}{\hat{\rho}(X)}$$

$$\hat{C}_\alpha(X) = [\hat{\mu}(X) \pm q_{1-\alpha}(S) \hat{\rho}(S)]$$

Full conformal prediction avoids data splitting it fit models \hat{A}_y on all the data plus (X_{n+1}, y) , for all possible values of y , and uses

$$S_y = \{s(X_i, Y_i, \hat{A}_y)\} \cup \{s(X_{n+1}, y, \hat{A}_y)\}$$

$$\hat{C}_\alpha(X_{n+1}) = \{y : s(X_{n+1}, y, \hat{A}_y) \leq q_{1-\alpha}(S_y)\}$$

The *jackknife* uses the leave-out-one residuals (but assumes the algorithm is “stable”).

$$S = \{|\hat{A}_{-i}(X_i) - Y_i|\} \cup \{\infty\}$$

$$\hat{C} = [\hat{A}(X_{n+1}) \pm q_{1-\alpha}(S)]$$

The *Jackknife+* does not make that assumption

$$s_i = |Y_i - \hat{\mu}_{-i}(X_i)|$$

$$u_i = \hat{\mu}_{-i}(X_i) + s_i$$

$$\ell_i = \hat{\mu}_{-i}(X_i) - s_i$$

$$U = \{u_i\} \cup \{\infty\}$$

$$L = \{\ell_i\} \cup \{\infty\}$$

$$C = [q_\alpha(L), q_{1-\alpha}(U)]$$

(the coverage is $\geq 1 - 2\alpha$, not $1 - \alpha$, unless the algorithm is stable).

Cross-Validation+ is between SCP and Jackknife+.

In a *covariate shift*, \mathcal{L}_X changes but not $\mathcal{L}_{Y|X}$. In a *label shift*, \mathcal{L}_Y changes, but not $\mathcal{L}_{X|Y}$. To deal with covariate shift, give more importance to calibration points closer to the test point.

$$(X_1, Y_1), \dots, (X_n, Y_n) \stackrel{\text{iid}}{\sim} P_X \times P_{Y|X}$$

$$(X_{n+1}, Y_{n+1}) \sim \tilde{P}_X \times P_{Y|X}$$

$$w_i \propto \frac{d\tilde{P}_X(x_i)}{dP_X(x)}$$

$$C_\alpha = \{y : s(X_{n+1}, y, \hat{A}) \leq q_{1-\alpha}(\sum w_i \delta_{s_i} + w_{n+1} \delta_\infty)\}$$

Use a similar idea for label shift.

For time series data, ACI (adaptive conformal inference) keeps track of the miscoverage ate (with an EWMA) and adjusts the quantile accordingly; AgACI uses several half-lives, combined with dynamic weights, and processes the upper and lower bounds separately.

**Convex analysis at infinity:
introduction to astral space
(ICML 2024 tutorial)**

The *astral space* is a compactification of \mathbf{R}^n for convex analysis.

We would like convex functions on \mathbf{R}^n to always have a minimum: to this end, we can add “points at infinity”. In dimension 1, we can just set $\bar{\mathbf{R}} = [-\infty, \infty]$ – the exponential function now has a minimum.

In dimension 2, we would like

$$f(x_1, x_2) = e^{-x_1} + (x_2 - x_1)^2$$

to have a minimum: we can add “points at infinity” for each direction and offset. But this is not sufficient: we also want

$$f(x_1, x_2) = e^{-x_1} + e^{-x_2+x_1/2}$$

to have a minimum, but the sequence $x_t = (t, t^2)$ does not correspond to a straight ray...

The *astral space* $\bar{\mathbf{R}}^n$ is obtained by adding to \mathbf{R}^n the sequences $(x_t)_t$ such that

$$\forall u \in \mathbf{R}^n \quad \lim(x_t \cdot u) \text{ exists (in } \bar{\mathbf{R}})$$

and identifying two sequences, $x_t \sim x'_t$ if $\forall u \quad \lim(x_t \cdot u) = \lim(x'_t \cdot u)$. This is a compact (but not metric) topological space.

For $\bar{x} \in \bar{\mathbf{R}}^n$, the *coupling function* is

$$\bar{x} \cdot u = \lim(x_t \cdot u).$$

It does not have all the properties of a scalar product, but

$$\bar{x} = \bar{x}' \text{ iff } \forall u \quad \bar{x} \cdot u = \bar{x}' \cdot u$$

$$\bar{x} \in \bar{\mathbf{R}}^n \text{ iff } \forall u \quad \bar{x} \cdot u \in \mathbf{R}$$

$$\bar{x}_t \rightarrow \bar{x} \text{ iff } \forall u \quad \bar{x}_t \cdot u \rightarrow \bar{x} \cdot u$$

An *astron* is an element of the form $x_t = tv$, for $v \in \mathbf{R}^n$ (a straight ray); it will be written $\omega v = \lim tv$. The elements of $\bar{\mathbf{R}}^2$ are of the form

$$x_t = t^2 v_1 + tv_2 + q, \quad v_1, v_2 \in \mathbf{R}^2$$

and are written $\bar{x} = \omega v_1 \star \omega v_2 \star q$ where \star is the *leftward addition*. On $\bar{\mathbf{R}}$:

$$(+\infty) \star (-\infty) = +\infty$$

$$(-\infty) \star (+\infty) = -\infty$$

$$x \star y = x + y \text{ otherwise.}$$

$$\text{On } \bar{\mathbf{R}}^n: (\bar{x} \star \bar{y}) \cdot u = (\bar{x} \cdot u) + (\bar{y} \cdot u).$$

In dimension n , every astral point can be written

$$\bar{x} = \underbrace{\omega v_1 \star \dots \star \omega v_k}_{\text{astrons}} \star \underbrace{q}_{\text{finite part}}$$

where v_1, \dots, v_k, q are orthogonal; k is the *astral rank* of \bar{x} .

A \mathcal{C}^0 function $f : \mathbf{R}^n \rightarrow \bar{\mathbf{R}}$ can be extended to $\bar{f} : \bar{\mathbf{R}}^n \rightarrow \mathbf{R}$ as

$$\bar{f}(\bar{x}) = \text{Min} \lim_{x_t \rightarrow \bar{x}} f(x_t).$$

For linear functions, $f(x) = x \cdot u$, the limit does not depend on the choice of the sequence, and $\bar{f}(\bar{x}) = \bar{x} \cdot u$, but, in general, \bar{f} is not continuous: for instance, with

$$f(x_1, x_2) = e^{-x_1} + e^{-x_2+x^2/2}$$

\bar{f} is minimized at $\bar{x} = \omega e_2 \star \omega e_1$, but \bar{f} is not continuous at \bar{x} : $x_t = (t, \frac{1}{2}t^2) \rightarrow \bar{x}$, but $f(x_t) \rightarrow 1 \neq 0$.

A convex function $f : \mathbf{R}^n \rightarrow \bar{\mathbf{R}}$ has a minimum at x iff $0 \in \partial f(x)$. This generalizes the subgradient at infinity:

$$u \in \partial \bar{f}(x) \text{ iff } \exists b \quad \forall \bar{y} \in \bar{\mathbf{R}}^n \quad \bar{f}(\bar{y}) \geq \bar{y} \cdot u + b \\ \exists x_t \rightarrow \bar{x} \quad f(x_t) - f(x_t \cdot u + b) \rightarrow 0$$

For constrained optimization

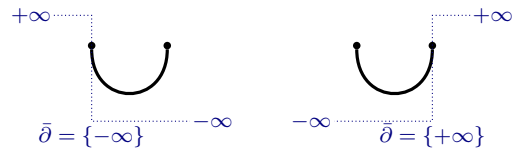
$$\text{Minimize}_{u \in \mathbf{R}^n} g(u) \quad \text{st } a \cdot u = b$$

a sufficient condition is $a \cdot u = b$, $\nabla g(u) = \lambda a$. We want to allow vertical tangents, as with

$$g(u) = \text{entropy}(u_1, u_2, 1 - u_1, -u_2)$$

(it is finite on the triangle, including its boundary, but it is not differentiable on the boundary – it is infinite outside the boundary). For $\bar{x} \in \bar{\mathbf{R}}^n$, the dual linear function is $\phi(u) = \bar{x} \cdot u$. The *dual subgradient* of a convex function $g : \mathbf{R}^n \rightarrow \bar{\mathbf{R}}$ is

$$\bar{x} \in \bar{\partial} g(u) \text{ iff } g(u) \in \mathbf{R} \\ \forall v \in \mathbf{R}^n \quad g(v) \geq \bar{x} \cdot (v - u) + g(u)$$



“Conjugacy inverts the subgradient map”: if f is closed, proper, convex,

$$u \in \partial f(x) \iff x \in \partial f^*(u).$$

More generally:

$$u \in \partial \bar{f}(\bar{x}) \iff \bar{x} \in \bar{\partial} f^*(u).$$

On \mathbf{R}^n , if f is convex, $\nabla f x = 0 \Rightarrow x$ is a minimizer. On $\bar{\mathbf{R}}^n$, if f is convex, $x_t \rightarrow \bar{x}$, and \bar{f} is continuous at

\bar{x} (this is not always the case, but most loss functions have a continuous extension), then

$$\nabla f(x_t) \rightarrow 0 \implies f(x_t) \rightarrow \inf f.$$

If f is convex and \bar{f} continuous, if the progress condition is satisfied

$$\begin{aligned} f(x_{t+1}) &\leq f(x_t) - \alpha_t h(\|u_t\|) \\ \alpha_t &\geq 0, \quad \sum \alpha_t = \infty \\ u_t &\in \partial f(x_t) \\ h &\text{ non-decreasing, } h(0) = 0, h(s) > 0 \end{aligned}$$

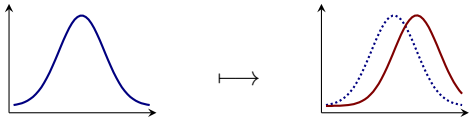
then $f(x_t) \rightarrow \inf f$.

Strategic ML: learning with data that behaves (ICML 2024 Tutorial)

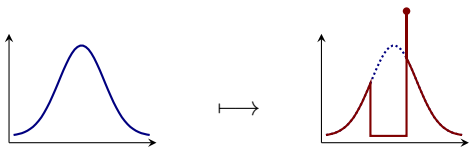
Strategic machine learning is a generalization of adversarial learning, in which subjects can change their features, to some extent. Assuming they know the model, it is a nested min-argmax problem (adversarial learning is a minimax problem).

$$\begin{array}{ll} \text{Find} & h, x' \\ \text{To minimize} & \mathbb{E} \mathbf{1}_{y \neq h(x')} \\ \text{Where} & x' = \underset{x'}{\operatorname{Argmax}} h(x') - \underset{\text{utility}}{c(x, x')} \end{array}$$

It is a Stackelberg game: the solution is an equilibrium. Adversarial learning assumes an (exogenous) distribution shift



while strategic learning uses a strategic shift.



If the subject is close to the decision boundary, it will move to be exactly on it, but if it is too far away, it will not move.

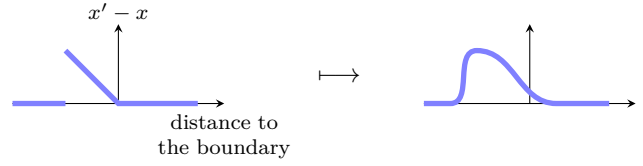
Replace the hinge loss with a strategic hinge



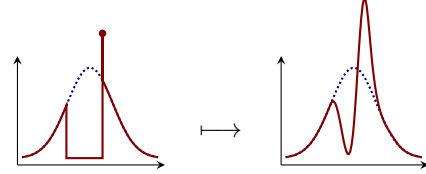
which penalizes points, not where they are, but where they could be.

Strategic ML is not necessarily adversarial: for instance, in a recommendation system, both users and the system want the user to be correctly classified.

If the users do not know the model exactly, their response is smooth.



The induced distribution is also smooth.



The price of opacity, $\text{POP} = \text{err}(h, \hat{h}) - \text{err}(h, h)$ where \hat{h} is the users' approximation if h , which can be arbitrarily bad.

Previously correctly classified subjects may need to change their features because the decision boundary shifted to limit gaming – but you may want to add a penalty for this cost (“social burden”).

Data attribution at scale (ICML 2024 tutorial)

1. There are several types of data attribution:

- Corroborative attribution (evidence finding) looks for evidence (citation, copyrighted work, etc.) in a corpus (reference data, which may not be part of the training set), for a given output: for instance, information retrieval;
- Game-theoretic attribution aims to understand why the model behaved in a given way, to assign fair credit or blame (liability) to different sources for the outcome (utility): for instance, “data Shapley”;
- Predictive attribution (data modeling) predicts the model output if we change the training data, estimating

$$\theta^*(w) = \underset{\theta}{\operatorname{Argmin}} \sum_i w_i \ell_i(\theta) \quad \text{for } w \neq \mathbf{1}_n.$$

2. Leave-out-one (LOO) has a closed form solution for linear regression; for logistic regression, we can use a quadratic approximation of $L_{-j}(\theta)$ around $\theta^*(\mathbf{1}_n)$. For nonlinear, but still convex, models, we can use a linear approximation

$$\text{LOO}(j) = \theta^* - \theta_{-j}^* \approx -\frac{d\theta}{dw_j}.$$

3. The influence function

$$\hat{f}(S \setminus \{z_j\}) = f(S) - \nabla \ell(\theta)^\top H^{-1} g_j$$

is too sensitive to hyperparameter choices. Instead, learn a mapping $\hat{f} : S \mapsto \hat{f}(S)$ maximizing

$$\text{LDS} = \text{Cor}[f(S^{(i)})_{1 \leq i \leq m}, \hat{f}(S^{(i)})_{1 \leq i \leq m}],$$

e.g.,

$$\hat{f}(S') = f(S) - \sum_{i \in S \setminus S'} \text{LOO}(i)$$

or

$$f(S) = \sum_{i \in S} \beta_i = \beta^\top \mathbf{1}_S.$$

4. Do not use older attribution methods: they have $\text{LDS} \approx 0$. Better influence function and Hessian approximations exist, e.g., Gauss-Newton

$$\begin{aligned} \tilde{H} + \lambda I \\ \tilde{H} &= (\nabla_\theta h)^\top \nabla_h^2 \ell(\nabla_\theta h) \approx \nabla_\theta^2 \ell \\ \ell &= \ell(h(x, \theta), y) \end{aligned}$$

or Kronecker-based approximations (EK-FAC). One can keep track of the training dynamics, estimating the impact of observation i at epoch t . One can also compute the attribution on a surrogate model

$$\text{Differentiable model} \xrightarrow{\text{NTK}} \text{High-dimensional linear model} \xrightarrow{\text{random projection}} \text{Low-dimensional linear model} \xrightarrow{\text{IF}} \hat{f}$$

Interpretable and explainable machine learning: a methods-centric overview with concrete examples
R. Marcinkevičs and J.E. Vogt (2023)

Explainable AI refers to methods to interpret (rationalize the decisions of) already trained black-box models: feature importance, Shapley values, saliency maps, integrated gradients, global surrogate models, local surrogate models (LIME), counterfactuals, etc.

Interpretable AI refers to models interpretable by construction – interpretability is provided by their “inductive bias”:

- Falling rule lists (with monotonic output as we move along the list);
- Supersparse linear integer models (SLIM): linear regression, with integer coefficients (possibly with a sign constraint) and binary inputs – the objective function uses a 0-1 loss, an ℓ^0 penalty, and a very small ℓ^1 penalty (for the ties from the ℓ^0 penalty); it is a MIP (after big-M transformation); you can approximate it with an integer-constrained lasso regression;
- GAM, GA²M (GAM with interactions), neural GAM, [GAM boosting];
- Sparse input neural nets (group lasso penalty for the first layer weights, to select which features to use);
- Knockoff features (replace x_i with random data sampled from the distribution of $x_i|x_{\setminus i}$ (ignoring y) to see how much value it adds to the model of interest);
- Varying coefficient models, self-explaining neural networks

$$f(x) = \theta(x)^\top x \quad (\text{or } f(x) = \theta(x)^\top h(x))$$

there θ is almost constant

$$\nabla_x f(x) \approx \theta(x_0) \quad \text{if } x \text{ is close to } x_0$$

(and θ may depend on only a subset of the inputs)

– Mixture of experts

$h_j(x)$: latent representation from expert j

$c_j(x)$: output from expert j

$u_j = \sigma(W_j \cdot [h_1, c_1, \dots, h_p, c_p] + b_j)$

$a_j \propto \exp\langle u_j, \theta_j \rangle$ attention weights

$$f(x) = \sum_j a_j c_j(x)$$

with an *auxiliary loss* to minimize the discrepancy between the prediction error if we remove j and the corresponding attention weight a_j ;

– Symbolic regression;

– Interpretable representation learning.

Designing inherently interpretable machine learning models
A. Sudjianto and A. Zhang (2021)

Checklist of qualitative characteristics an interpretable model should have: additivity, sparsity, linearity, smoothness, monotonicity, visualizability, projection (sparse or near-orthogonal projections are more interpretable), segmentation (models with few “segments” are more interpretable).

GAMI-Net: an explainable neural network based on generalized additive models with structured interactions
Z. Yang et al.

GA²M,

$$y \approx \sum_i g_i(x) + \sum_{ij} g_{ij}(x_i, x_j)$$

with neural networks, sparsity penalty, and “marginal clarity” penalty ($g_i \perp\!\!\!\perp g_{ij}$).

Granger-causal attentive mixtures of experts: learning important features with neural nets
P. Schwab et al. (2019)

Define a GAM-VCM (variable coefficient model)

$$f(x) = \sum_i w_i(x) h_i(x)$$

as follows:

- Transform each input feature x_i in a nonlinear way into a forecast (“contribution”) c_i , with an MLP, with activations (latent representation) h_i

$$\begin{array}{ccccc} x_i & \longmapsto & h_i & \longmapsto & c_i \\ \text{(scalar)} & & & & \text{(scalar)} \end{array}$$

- Concatenate all the latent representations h_i and forecasts c_i into a vector h ;
- For each “expert” i , compute a context vector u_i from h , and compare it with a learned reference context u_i^{ref} , with a scalar product, $\langle u_i, u_i^{\text{ref}} \rangle$
- Compute the corresponding weights with a softmax,

$$w_i \propto \langle u_i, u_i^{\text{ref}} \rangle$$

(the paper has a different formula).

To force those “attention” weights to reflect feature importance, add a penalty, computed as follows:

- Train auxiliary predictors, using h and $h_{\setminus i}$: $p(h)$, $p_i(h_{\setminus i})$;
- Compute their error

$$\begin{aligned}\varepsilon &= \text{loss}(y, p(h)) \\ \varepsilon_i &= \text{loss}(y, p_i(h_{\setminus i})) \\ \Delta\varepsilon_i &= \varepsilon_i - \varepsilon \geq 0;\end{aligned}$$

- Compare those errors with the attention weights

$$\begin{aligned}\omega_i &= \frac{\Delta\varepsilon_i}{\sum_j \Delta\varepsilon_j} \\ \text{KL}(\omega||w) &= \sum \omega_i \log \frac{\omega_i}{w_i}\end{aligned}$$

(w_i depends on the sample, ω_i does not: average over all samples).

There is no penalty to make w slow-moving.

Self-explaining neural networks **D. Alvarez-Melis and T.S. Jaakola**

Variable coefficient models (VCM) are of the form

$$f(x) = \sum_i \theta_i(x) x_i$$

where the θ_i ’s are slow-moving functions (often, of a subset of the x_i ’s). They can be generalized to

$$f(x) = \sum_i \theta_i(x) h_i(x)$$

where h is a (low-dimensional) representation of the data (e.g., from an auto-encoder). If θ is almost constant,

$$\begin{aligned}\nabla_x f &= \theta(x)^\top \nabla_x h + \nabla_x \theta^\top h(x) \\ &\approx \theta(x)^\top \nabla_x h;\end{aligned}$$

the model can be trained by adding a penalty $\|\nabla_x f - \theta(x)^\top \nabla_x h\|$.

Towards robust interpretability with self-explaining neural networks **D. Alvarez-Melis and T.S. Jaakola**

PiML toolbox for interpretable machine learning model development and diagnostics **A. Sudjianto et al. (2023)**

Closed-source; check `imodels` instead.

Varying coefficient models **T. Hastie and R. Tibschirani (1993)**

Variable coefficient models are locally linear:

$$y \approx \sum_j \beta_j(x) x_j.$$

Often, the coefficients β_j only depend on a subset R_j of the predictors: $y \approx \sum_j \beta_j(R_j) x_j$. They can be estimated as

$$\beta_j(R_j) = \frac{\text{E} \left[X_j^2 \frac{Y - \sum_{k \neq j} \beta_k(R_k) X_k}{X_j} \middle| R_j \right]}{\text{E}[X_j^2 | R_j]}$$

with tensor product splines, minimizing

$$\left[\sum_i y_i - \sum_j x_{ij} \beta_j(r_{ij}) \right]^2 + \sum_j \lambda_j \int \beta_j''(r_j) dr_j$$

Sparse-input neural networks for high-dimensional nonparametric regression and classification **J. Feng and N. Simon (2019)**

Add a sparse group lasso penalty on the first layer weights W_1 .

$$\text{loss} = \text{error} + \lambda_0 \sum_{a \geq 2} \|W_a\|_2^2 + \lambda_{12} \|W_1\|_{12} + \lambda_1 \|W_1\|_1$$

Gradient descent (or other standard gradient-based optimizers) will not give a sparse solution: use proximal gradient.

$$\begin{aligned}W &\leftarrow w - \gamma \nabla \text{loss}_{\text{smooth}} \\ W_1 &\leftarrow S(w_1, \gamma \lambda_1) \\ W_{1:i} &\leftarrow \left(1 - \frac{\gamma \lambda_{12}}{\|W_{1:i}\|_2} \right)_+ W_{1:i}\end{aligned}$$

Supersparse linear integer models for interpretable classification **B. Ustun et al. (2014)**

For interpretable linear models, SLIM uses:

- Binary predictors,
- 0-1 loss;
- ℓ_0 penalty,
- Very small ℓ_1 penalty to avoid the ties created by the ℓ_0 penalty;
- Integer coefficients (or number with just one significant digit, possibly with a sign constraint).

This can be solved as a mixed integer linear program (with a big- M transformation), and approximated with an integer-constrained lasso regression.

Supersparse linear integer models for optimized medical scoring systems **B. Ustun and C. Rudin**

***Interpretable machine learning
based on functional Anova framework:
algorithms and comparison***
L. Hu et al. (2023)

There are many generalizations of GAM:

$$\begin{array}{ll} \text{GAM} & g(x) = g_1(x_1) + \cdots + g_n(x_n) \\ \text{AIM} & g(x) = g_1(\beta'_1 x) + \cdots + g_n(\beta'_n x) \\ \text{GA}^2\text{M} & g(x) = \sum g_i(x_i) + \sum g_{ij}(x_i, x_j) \end{array}$$

(GA²M is also known as fANOVA).

Explainable boosting machines (EBM) are similar to GA²M, but the g_i 's and g_{ij} 's are decision trees: one first fits the first order effects g_i , and only then the second order effects g_{ij} on the residuals, but only for the top k interactions, identified by comparing 4 quadrants.

GAMI-Lin-T (GAM with interactions and linear trees) are similar to EBM, but:

- The leaves contain linear functions;
- For the interactions, one variable is used only for splitting, and the other one is only used in the linear models, so we have two different terms g_{ij} and g_{ji} ;
- The top- k interactions are selected by comparing the predictive power (on the residuals) of the model trees;
- There is an additional “purifying” step, to ensure orthogonality.

$$\begin{aligned} \langle g_{ij}(x_i, x_j), g_i(x_i) \rangle &= 0 \\ \langle g_{ij}(x_i, x_j), g_j(x_j) \rangle &= 0 \end{aligned}$$

GAMINet is similar but uses neural nets for g_i and g_{ij} , with a “clarity” penalty to ensure orthogonality

$$\Omega_{jk} = \left| \frac{1}{N} \sum_{i \in \text{Observations}} g_j(x_{ij}) g_{jk}(x_{ij}, x_{ik}) \right|$$

(and a monotonicity penalty if desired).

***Using model-based trees with boosting
to fit low-order functional Anova models***
L. Hu et al.

Details of the boosting algorithm and the interaction filtering method.

***Explainable neural networks
based on additive index models***
J. Vaughan et al. (2018)

AIMs can be estimated with back-fitting or end-to-end (xNN): projection followed by univariate transformations (neural nets)

Adaptive explainable neural networks (AxNNs)
J. Chen et al. (2020)

Ensemble (boosting) of GAM and AIM, both implemented as neural nets.

$$\begin{array}{ll} \text{GAM:} & g(x) = \sum_i g_i(x_i) \\ \text{AIM:} & g(x) = \sum_i g_i(\beta'_i x) \end{array}$$

***Linear iterative feature embedding:
an ensemble framework for interpretable model***
A. Sudjianto et al.

A wide, single-layer neural net can be seen as an ensemble of narrow single-layer nets:

- Take k random neural nets

$$\begin{cases} \mathbf{R}^d & \longrightarrow \mathbf{R} \\ x & \longmapsto \hat{y} \end{cases}$$

- Use them to define samples of the data $\{i : \hat{y} > a\}$ (or $\hat{y} < a$); discard samples too large or too small;
- Train new neural nets on those samples (to learn features);
- Iterate a few times
- Combine the features thus learned (elastic net).

***An optimization approach
to learning falling rule lists***
C. Chen and C. Rudin (2018)

Monte Carlo search on frequent itemsets.

***Counterexample guided learning
of monotonic neural networks***
A. Sivaraman et al.

For each input sample x , find (with an SMT solver) the worst monotonicity counterexamples in each direction:

$$\begin{aligned} x_\ell &\geq x \quad \text{with} \quad f(x_\ell) \leq f(x) \\ x_u &\leq x \quad \text{with} \quad f(x_u) \geq f(x) \end{aligned}$$

These define the lower and upper envelopes of f .

Use them during training: (every other iteration) replace each sample (x_i, y_i) with (x_i, \hat{y}_i) , $(x_{i\ell}, \hat{y}_i)$, (x_{iu}, \hat{y}_i) , where $\hat{y}_i = \frac{1}{2}[f(x_{i\ell}) + f(x_{iu})]$.

***Interpretable machine learning
with PySR and SymbolicRegression.jl***
M. Cranmer (2023)

***Evaluating the visualization
of what a deep neural network has learned***
W. Samek et al.

Layerwise relevance propagation (LRP) gives, for each layer, a decomposition of the network output (e.g., the

log-odds, for a classification problem) into contributions of the individual neurons of that layer; in particular, the sum of the contributions is the same for all layers:

$$\sum_i R_i^{(\ell)} = \sum_j R_j^{(\ell+1)}.$$

The contributions in layer ℓ can be computed from those in layer $\ell + 1$, for instance as

$$R_i^{(\ell)} = \sum_j \left(\alpha \frac{z_{ij}^+}{\sum_{i'} z_{i'j}^+} + \beta \frac{z_{ij}^-}{\sum_{i'} z_{i'j}^-} \right) R_j^{(\ell+1)}$$

$$z_{ij} = w_{ij}^{(\ell \rightarrow \ell+1)} a_i^{(\ell)}$$

$$z_{ij} = z_{ij}^+ + z_{ij}^-$$

$$\alpha + \beta = 1.$$

Layerwise relevance propagation: an overview
G. Montavon et al.

There are other propagation rules.

***On pixelwise explanations
for nonlinear classifier decisions
by layer-wise relevance propagation***
S. Bach et al. (2015)

Original paper on LRP.

***On inductive biases for machine learning
in data-constrained settings***
G. Mialon (2022)

To approximate a kernel K , use

$$K(x, x') \approx \langle \psi(x), \psi(x') \rangle$$

$$\psi(x) = K(Z, Z)^{-1/2} K(Z, x)$$

where Z is a set of “anchoring points”.

***A trainable optimal transport embedding
for feature aggregation and
its relationship to attention***
G. Mialon et al. (2021)

To get a representation of (varying length) sequences x , encoded as sets of k -mers with positional embeddings, compute the optimal transport P (wrt some kernel) to a learned fixed-length sequence z , and use the optimal transport as weights: $(x_i)_i \mapsto (\sum_j P_{ij} z_j)_j$.

You could use several reference sequences. This is a pooling operation.

***GraphiT:
encoding graph structure in transformers***
G. Mialon et al. (2021)

To use transformers with graphs:

- Use a masked transformer (so that each node only attends to its k -hop neighbours) – or not;

- Use the eigenvectors of the Laplacian as positional encoding, $L = \sum \lambda_i u_i u_i'$;
- Use a graph kernel to compute the positional encoding;

$$k_r = \sum r(\lambda) u_i u_i'$$

$$r(\lambda) = e^{-\beta \lambda} \quad \text{diffusion}$$

$$r(\lambda) = (1 - \gamma \lambda)^p \quad p\text{-step random walk}$$

- Rescale the attention weights: $\exp(QQ'/\sqrt{d}) \odot k_r$.

***Screening data points
in empirical risk minimization
via ellipsoidal regions and safe loss functions***
G. Mialon et al. (2020)

$$f^*(y) = \text{Max}_t \langle t, y \rangle - f(t) \quad \text{Fenchel conjugate}$$

$$f^{**}(t) = \text{Max}_y \langle y, t \rangle - f^*(y) \quad \text{biconjugate}$$

$$f_\mu(t) = \text{Max}_y \langle y, t \rangle - f^*(y) - \mu \Omega(y)$$

$$= \text{Min}_z f(z) + \mu \Omega^* \left(\frac{t - z}{\mu} \right)$$

$$= t \square_\mu \Omega^* \quad \text{infimum convolution}$$

***safeaipackage: a Python package
for AI risk measurement***
G. Babaei

Given a (nonnegative) random variable Y :

- The Lorenz curve is `(ts, cumsum(Y[i]))`, where `ts=linspace(0,1,len(Y))`, `i=argsort(Y)`;
- The dual Lorenz curve uses `i=argsort(-Y)`;
- The concordance curve uses `i=argsort(y')` where y' is another variable.

The area under the concordance curve is

$$\frac{1}{2} \frac{\text{Cov}[Y[i'], F(Y[i])]}{\text{Cov}[Y[i], F(Y[i])]} - \frac{1}{2} \in [0, 1]$$

Different choices of (Y, Y') give different measures:

- Accuracy (“RGA”): (y, \hat{y}) (for binary data, this is the AUC);
- Robustness: (\hat{y}, \hat{y}^p) , where \hat{y}^p are forecasts after some perturbation;
- Explainability: (\hat{y}, \hat{y}^{-k}) , where \hat{y}^{-k} is the forecast of a model fitted without variable k ;
- Fairness: (\hat{y}, \hat{y}^{-k}) , where k is some protected attribute;
- Privacy: (\hat{y}, \hat{y}^{-m}) , where \hat{y}^{-k} are forecasts of a model fitted without observation m .

RGA: a unified measure of predictive accuracy
P. Giudici and E. Raffinetti (2022)

***A rank graduation accuracy measure
to mitigate artificial intelligence risks***
E. Raffinetti (2023)

Pretaining and the lasso

E. Craig et al.

Pretrain the lasso on a large dataset

$$y = C\beta'_1 + \varepsilon \quad \beta_1 \text{ sparse}$$

then, fine-tune it on the subset of interest (stratification) or on a different, smaller dataset

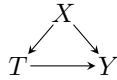
$$y - X\beta'_1 = X\beta'_2 + \eta \quad \beta_2 \text{ sparse}$$

where the lhs, the residuals of the first model, can be replaced by $y - (1 - \alpha)X\beta'_1$.

Estimation and inference of heterogeneous treatment effects using random forests

S. Wager and S. Athey (2017)

Under unconfoundedness $\{Y_i^{(0)}, Y_i^{(1)}\} \perp\!\!\!\perp T_i \mid X_i$, i.e., X are the only confounders



the treatment effect $\tau(X) = E[Y_i^{(1)} - Y_i^{(0)} \mid X_i = x]$ can be computed as

$$\tau(x) = E \left[Y_i \left(\frac{T_i}{e(x)} - \frac{1 - T_i}{1 - e(x)} \right) \mid X_i = x \right]$$

where $e(x) = E[T_i \mid X_i = x]$ is the propensity of receiving the treatment.

Causal forests directly estimate τ without estimating e first, by noticing that if the leaves of an (honest) tree are small enough, they look like randomized experiments and the treatment effect can be estimated with unweighted averages

$$\hat{\tau}(x) = \text{Mean}_{\substack{i \in L \\ T_i=1}} Y_i - \text{Mean}_{\substack{i \in L \\ T_i=0}} Y_i$$

where L is the leaf containing x .

There are two ways of building honest trees:

- A *propensity tree* is built without using the outcome Y , to predict the treatment T , ensuring that each leaf contains at least k observations for each treatment;
- A *double sample tree* is built using half the data, to forecast the outcome Y , but the responses are computed using the other half.

HEBO: Heteroskedastic evolutionary Bayesian optimization

A.I. Cowen-Rivers et al.

For Bayesian optimization on heteroskedastic non-stationary data:

- Use an input-warped $\Psi(x) = 1 - (1 - x^a)^{b-1}$ GP (linear + Matérn 3/2, from GPy)
- Power-transform (Box-Cox or Yeo-Johnson, in `sk-learn`) the output;
- Add noise to the posterior mean;
- Use multiple acquisition functions (NSGA-II, from `pymoo`).

Unsupervised learning of visual features by contrasting cluster assignments

M. Caron et al.

SwAV computes image features by minimizing $\ell(z_t, q_s) + \ell(z_s, q_t)$, where

- x_t and x_s are augmentations of the same image;
- $z = f_\theta(x) / \|f_\theta(x)\|$ are the corresponding features;
- c_1, \dots, c_k are (trainable) prototype vectors;
- $q = \underset{c \in \{c_1, \dots, c_k\}}{\text{Argmax}} \langle c, z \rangle$;

The three types of backtests

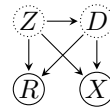
J. Joubert et al. (2024)

Historical, cross-validation, Monte Carlo (if you have one (or several) data generation process(es) for *all* the data used).

Gene regulatory network inference in the presence of dropouts: a causal view

H. Dai et al. (2024)

scRNAseq data contains biological zeroes (no gene expression) and technical zeroes (“dropouts”). Since $Z_i \perp\!\!\!\perp Z_j \iff X_i \perp\!\!\!\perp X_j \mid R = 0$, we can just drop the zeroes (dropouts and biological zeroes) and recover the correct conditional independence relations.



Z : True expression

D : $\mathbf{1}_{\text{dropout}}$

X : observed expression

R : observed zero

Robust agents learn causal world models

J. Richens and T. Everitt (2024)

If an RL agent is robust to distributional shifts, we can extract a causal graph from it.

Predictive auxiliary objectives in deep RL mimic learning in the brain

C. Fang and K. Stachenfeld (2024)

Auxiliary objectives in RL systems, for (long-horizon) latent state forecast (contrastive prediction), help representation learning (prevents representation collapse).

ASID: active exploration for system identification in robotic manipulation

M. Memmel et al. (2024)

Model-free RL is sample-inefficient, but model-based RL requires an accurate world simulator:

- Start with an inaccurate simulator (correct model, with approximate parameters);

- Train an exploring agent, maximizing Fisher information

$$\text{Minimize}_{\pi} \text{trace } I(p_{\theta^*}(\cdot|\pi))$$

(where p is the distribution on trajectories);

- Collect real-world data;
- Use it to refine the simulator;
- Train the (exploiting) agent.

Learning interactive real-world simulators
M. Yang et al. (2024)

UniSim is a UNet diffusion model trained on a variety of labeled video datasets, to forecast future frames from past frames and (natural language) “actions”, for reinforcement learning.

Semantic compression with large language models
H. Gilbert et al. (2024)

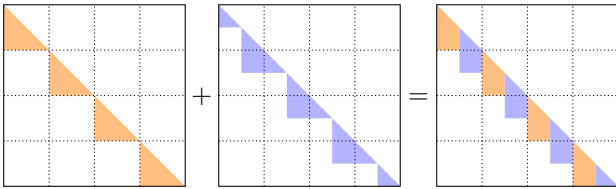
Ask an LLM to compress text; the result does not have to be human readable, but another GPT-4 model should be able to uncompress it.

LoftQ: LoRA-fine-tuning-aware quantization for large language models
Y. Li et al. (2023)

When fine-tuning with LoRA after quantization, do not initialize the LoRA weights with zeroes, but have them approximate the unquantized model.

LongLoRA: efficient fine-tuning of long-context large language models
Y. Chen et al.

Shifted sparse attention increases the context size of a transformer by using full attention on overlapping blocks.



Do not only use LoRA for the attention weights, also fine-tune the embeddings and the normalization.

Batched low-rank adaptation of foundation models
Y. Wen and S. Chaudhuri (2024)

Input-specific LoRA weights, for personalized, task-specific adaptation, by replacing $W + BA'$ with $W \odot (BA')$

Models tell you what to discard: adaptive KV cache compression for LLMs
S. Ge et al. (2024)

The KV cache mechanism stores previous key-value vectors to avoid recomputing them each time. Recognize the (head-specific) attention patterns:

- Local context; special tokens (punctuation, etc.);
- Column-sparse;
- Dense.

and discard unneeded tokens.

Vision transformers need registers
T. Darcet et al.

Attention maps of ViT are not informative: tokens in low-information regions (e.g., sky) appear important – they are used for internal computations. Provide additional tokens (“registers”), neither in the input nor in the output (contrary to the CLS token): the attention maps become interpretable.

Beyond Weisfeiler-Lehman: a quantitative framework for GNN expressiveness
B. Zhang et al. (2024)

Given a GNN model \mathcal{M} , computing graph representations $G \mapsto \mathcal{M}(G)$, and a substructure (a graph) F , \mathcal{M} can count F under homomorphism if

$$\forall G, H \mathcal{M}(G) = \mathcal{M}(H) \Leftrightarrow |\text{hom}(F, G)| = |\text{hom}(F, H)|.$$

The homomorphism expressivity of \mathcal{M} is the set $\mathcal{F}(\mathcal{M})$ of all substructures it can count. For instance:

$$\mathcal{F}(\text{MPNN}) = \{\text{forests}\}$$

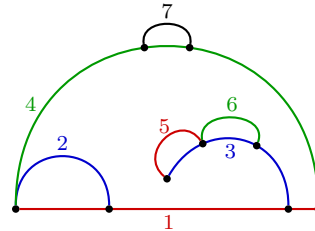
$$\mathcal{F}(\text{subgraph GNN}) = \{F : F \text{ has an end-point-shared NED}\}$$

$$\mathcal{F}(\text{local 2-GNN}) = \{F : F \text{ has a strong NED}\}$$

$$\mathcal{F}(\text{local 2-FGNN}) = \{F : F \text{ has an almost strong NED}\}$$

$$\mathcal{F}(\text{2-FGNN}) = \{F : F \text{ has a NED}\}$$

A *nested ear decomposition* (NED) is:



A *subgraph GNN* applies a MPNN to all marked subgraphs $\{\{G^u : u \in V_G\}\}$.

GNNCert: deterministic certification of graph neural networks against adversarial perturbations
Z. Xia et al.

Divide the graph (the set of edges and/or features) into (non-overlapping) subgraphs; use a graph classifier on

each of them, and take the majority vote: this is robust to a bounded number of edge or feature perturbations.

Graph neural networks for learning equivariant representations of neural networks

M. Kofinas et al.

Neural networks are computation graphs: they can be processed with GNNs or transformers, which automatically account for their permutation symmetries.

Applications:

- Implicit neural representation (INR) classification;
- Predicting the generalization performance of a CNN classifier (example: models from the small CNN zoo – same architecture (not required), different weights);
- Improving the weights of a neural net (learning to optimize), given weights, gradients, and momentum, at several scales.

Self-RAG: learning to retrieve, generate, and critique through self-reflection

A. Asai et al.

Ask the LLM if they need RAG, if the retrieved documents are relevant, if the generated answer is consistent with these documents. Use special tokens, RETRIEVE, ISRELEVANT, SUPPORTED, etc. (ask an LLM to add those tokens to the data).

Knowledge card: filling LLM’s knowledge gaps with plugin specialized language models

S. Feng et al. (2024)

Train several small, domain-specific LLMs, and use the text they generate instead of RAG:

- Bottom-up: generate text from the query for all of them, refine those texts, and ask an LLM to use them to answer the query;
- Top-down: first, ask the LLM if it needs more data, and which specialized LLMs to use.

The documents generated by those specialized LLMs are preprocessed for relevance, brevity and factuality:

- Check relevance with the cosine similarity with the query, only keeping the top k ;
- Ensure brevity by summarizing the documents
- Check the summary is consistent with the original text;
- Check that it is factually correct using RAG.

Self-alignment with instruction back-translation

X. Li et al.

To generate instructions to fine-tune an LLM, start with the answers (good quality web pages) and ask the LLM to produce instructions that would lead to those answers; also ask the LLM to select high-quality examples.

Amortizing intractable inference in large language models

E.J. Hu et al.

Given an LLM, we may want to sample from related distributions, e.g.:

- Infilling: $q(B|A, C) \propto p_{\text{LM}}(ABC)$;
- Tempered sampling: $q(B|A) \propto p_{\text{LM}}(AB)^{1/T}$;
- Constrained generation: $q(A) \propto p_{\text{LM}}(A)c(A)$.

This can be done with *GFlowNets*, a diversity-seeking reinforcement learning algorithm, training a policy to sample tokens from an unnormalized density.

Improved techniques for training consistency models

Y. Song and P. Dhariwal (2024)

Diffusion models learn a denoising function

$$(x_{\sigma+\Delta\sigma}, \sigma + \Delta\sigma) \mapsto x_{\sigma}.$$

Consistency models directly learn

$$f_{\theta} : (x_{\sigma}, \sigma) \mapsto x_0$$

by minimizing (a weighted expectation of)

$$d[f_{\theta}(x_{\sigma_{i+1}}), f_{\theta}(\check{x}_{\sigma_i})]$$

where $\theta^- = \text{EWMA}(\theta)$, $\check{x} = x - \sigma \cdot \Delta\sigma \cdot \nabla_x \log p_{\sigma}(x)$ and the score function $\nabla_x \log p_{\sigma}(x)$ comes from a diffusion model (consistency distillation). Instead, consistency training uses $\check{x} = x + \sigma z$, $z \sim N(0, I)$.

Würstchen: an efficient architecture for large-scale text-to-image diffusion models

P. Pertinas et al. (2024)

3-stage latent diffusion model (LDM):

- A first LDM computes a very low-dimensional latent representation of the image, conditioned on the input text;
- A second LDM computes a low-dimensional latent representation of the image, conditioned on the output of the previous stage;
- Finally, a VQGAN decoder generates the full-resolution image.

Lipschitz singularities in diffusion models

Z. Yang et al.

Diffusion models are not Lipschitz at $\sigma = 0$. E-TSDM replaces $f : (x_t, t) \mapsto x_{t-\Delta t}$ with

$$\tilde{f} : (x_t, t) \mapsto \begin{cases} f(x_t, t) & \text{if } t > \tilde{t} \\ f(x_t, \tilde{t}) & \text{if } t \leq \tilde{t} \end{cases}$$

Generalization in diffusion models arises from geometry-adaptive harmonic representations
Z. Kadkhodaie et al.

Diffusion models trained on sufficiently large (10^5) non-overlapping datasets learn the same score function and generate similar images: they do not memorize their training data.

They have a GAHB bias. A (bias-free) denoiser is locally linear: $f(y) \approx \nabla f \cdot y$; its Jacobian is approximately symmetric (and psd), so it has an eigen decomposition $f(y) \approx \sum_k \lambda_k(y) \langle y, e_k(y) \rangle e_k(y)$. The denoiser performs some shrinkage in that (data-dependent) basis; this basis shows oscillating patterns.

Improving convergence and generalization using parameter symmetries
B. Zhao et al.

In the optimization problem

$$\text{Minimize}_{g \in \mathbf{R}^d} \ell(w),$$

if the loss ℓ is invariant under the action of some group G ,

$$\forall g \in G \quad \forall w \in \mathbf{R}^d \quad \ell(g \cdot w) = \ell(w),$$

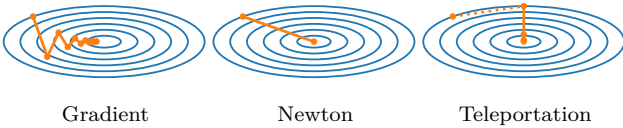
symmetry teleportation is a variant of gradient descent looking for the point with the steepest gradient in the current orbit,

$$w \leftarrow g \cdot w$$

$$g = \underset{g}{\text{Argmax}} |\nabla \ell(g \cdot w)|$$

Teleporting towards larger curvatures may improve generalization.

Symmetry teleportation for accelerated optimization
B. Zhao et al.



Linear neural nets are invariant under an action of GL_{d_m} :

$$g \cdot W_m = W_m g^{-1}$$

$$g \cdot W_{m-1} = g W_{m-1}$$

$$g \cdot W_k = W_k \text{ if } k \notin \{m, m-1\}.$$

This can be generalized to neural nets with invertible nonlinearities (e.g., leaky ReLU),

$$g \cdot W_{m-1} = \sigma^{-1} [g \sigma(W_{m-1} h_{m-2})] h_{m-2}^{-1}$$

$$h_{m-1} \mapsto W_m h_{m-1} \mapsto \sigma(W_m h_{m-1})$$

(if h_{m-2} is square and invertible); note that the action depends on the data.

Symmetries, flat minima, and the conserved quantities of gradient flow
B. Zhao et al.

Given a 2-layer neural net

$$\begin{matrix} x & \longrightarrow & \boxed{V} & \longrightarrow & \boxed{\sigma} & \longrightarrow & \boxed{U} & \longrightarrow & F(x) \\ \mathbf{R}^n & & & & \mathbf{R}^h & & & & \mathbf{R}^m \end{matrix}$$

(for which $\forall z \sigma(z) \neq 0$, e.g., $\sigma = \text{sigmoid}$), there is a (data-dependent) action of GL_h on $\text{Param} \times \mathbf{R}^n$

$$g \cdot (U, V, x) = (U R_{\sigma(Vx)} R_{\sigma(gVx)}^{-1}, gV, x)$$

where

$$(R_z)_{ij} = \begin{cases} z_i \cos(\alpha_{j-1}) \left(\prod_{k < j} \sin \alpha_k \right) & \text{if } j \leq i \\ -r \sin \alpha_i & \text{if } j = i + 1 \\ 0 & \text{otherwise} \end{cases}$$

and $(r, \alpha_1, \dots, \alpha_{h-1})$ are the spherical coordinates of z

$$z_i = r \cos \alpha_i \prod_{k < i} \sin \alpha_k.$$

InfoBatch: lossless training speed up by unbiased dynamic data pruning
Z. Qin et al.

Data pruning (filtering out samples contributing little to training) leads to biased gradients. Use dynamic data pruning instead: keep track of the loss of each sample, randomly drop some proportion of the low-loss samples (anew for each epoch) and rescale the gradients of the remaining low-loss samples; for the last few epochs, use the whole dataset.

Towards a statistical theory of data selection under weak supervision
G. Kolossov et al.

To select on which data to train:

$$\pi_i = \text{Min}\{c \cdot (\text{Var } \hat{Y}_i)^\alpha; 1\}$$

$$w_i = 1 \text{ or } 1/\pi_i$$

where

- Fit a model on a small subset of the data and compute \hat{Y}_i from this surrogate model (better surrogates need not be useful);
- $\alpha < 0$ prefers easy samples (better if you want a small number of samples);
- $\alpha > 0$ prefers hard samples;
- c is chosen to have the desired number of samples;
- $w_i = 1/\pi_1$ leads to unbiased (but possibly worse) results.

Use cross-validation to choose the surrogate model, α and w – it will depend on the dimensionality of the problem, on the model, and on the proportion of the data you want to keep.

**Meta continual learning revisited:
implicitly enhancing
online Hessian approximation
via variance reduction**
Y. Wu et al.

Regularization-based continual learning uses updates of the form

$$\theta \leftarrow \theta - \alpha(H^1 + \dots + H^{k-1})^{-1} \nabla_{\theta} \mathcal{L}^k(\theta)$$

where H^i is the Hessian of the loss at the end of training for task i (e.g., the diagonal Fisher information matrix) to keep the weights that mattered for task i .

**Topological data analysis
on noisy quantum computers**
I.Y. Akhalwaya et al.

Quantum computers do linear algebra in exponentially large vector spaces. This is what TDA needs (computing the rank of an exponentially large matrix).

**Cameras as rays:
pose estimation via ray diffusion**
J.Y. Zhang et al.

A camera is typically parametrized as $(R, t, k) \in \text{SO}(3) \times \mathbf{R}^3 \times \mathbf{R}^{3 \times 3}$

$$\begin{cases} \text{World} & \longrightarrow \text{Pixels} \\ x & \longmapsto u = k[R|T]x. \end{cases}$$

Instead, model a camera as a collection of rays (r_1, \dots, r_m) associated to pixel coordinates (u_1, \dots, u_m) (patches). The ray in direction $d \in \mathbf{R}^3$ through point $p \in \mathbf{R}^3$ can be represented with Plücker coordinates

$$(d, m = p \times d) \in \mathbf{R}^3 \times \mathbf{R}^3$$

(actually $\mathbf{P}(\mathbf{R}^6)$: the Plücker embedding is $\text{Gr}_k V \hookrightarrow \mathbf{P}(\Lambda^k V)$, here with $k = 2$, $\dim V = 4$ – lines in 3-dimensional projective space are 2-dimensional linear subspaces in 4-dimensional space). To convert from cameras to rays:

$$\begin{aligned} d &= R'K^{-1}u \\ m &= (-R't) \times d \end{aligned}$$

To convert from rays to cameras: first, find the camera center, $c = \text{Argmin}_p \sum \|p \times d - m\|^2$, then the camera transformation $P = \text{Argmin}_{\|H\|=1} \sum \|Hd \times u\|$. Train a transformer (with positional encoding) to predict rays from patch features or, better if there is ambiguity because the data is too sparse, a diffusion model.

**Accelerating distributed stochastic
optimization via self-repellent random walks**
J. Hu et al.

Given a time-reversible Markov chain with transition kernel P and stationary distribution μ , the *self-repellent random walk* (SRRW)

$$K_{ij}(x) \propto P_{ij} \left(\frac{x_i}{\mu_j} \right)^{-\alpha},$$

where x_j is the number of visits to state j , has a smaller asymptotic variance and converges faster.

The *token algorithm* is a decentralized SGD, implemented using a random walk (e.g., SRRW) on a graph, each node updating θ using its local gradient and sending the new value to one of its neighbours.

**Self-repellent random walks on general graphs:
achieving minimal sampling variance
via nonlinear Markov chains**
V. Doshi et al. (2023)

Stochastic heavy ball
S. Gadat et al. (2016)

To find the minimum of a function f , the (deterministic) *heavy ball* method (HBF, heavy ball with friction) considers a ball, moving on the graph of f , subject to both damping γ and acceleration

$$\ddot{x}_t + \gamma_t \dot{x}_t + \nabla f(x_t) = 0.$$

It converges towards a minimum of f if

$$\begin{aligned} \int_0^\infty \gamma_s ds &= \infty \\ \int_0^\infty \exp\left(-\int_0^t \gamma_s ds\right) dt &< \infty, \end{aligned}$$

e.g., if $\gamma_t = r/t$ ($r > 1$) or $\gamma_t = \gamma > 0$.

It can be discretized

$$\begin{aligned} y_{t+1} &= (1 - \gamma_t)y_t - \nabla f(x_t) \\ x_{t+1} &= x_t + y_{t+1} \end{aligned}$$

and one can replace the gradient $\nabla f(x_t)$ with an approximation (computed on a different minibatch each time).

**Understanding in-context learning
in transformers and LLMs
by learning to learn discrete functions**
S. Bhattamishra et al.

In-context learning (ICL) is the task of forecasting y from x after seeing a handful of examples.

$$(x_1, y_1, x_2, y_2, \dots, x_n, y_n, x_{n+1}) \mapsto y_{n+1}$$

**ReLU strikes back: exploiting activation
sparsity in large language models**
I. Mirazadeh et al.

ReLU (contrary to the trendier GELU or SiLU) activation functions lead to sparse activations, and a significant reduction of CPU-GPU data transfers, with a negligible impact on convergence and performance. For pretrained models, replace their activation functions with ReLU and fine-tune them.

**Unprocessing seven years
of algorithmic fairness**
A.F. Cruz and M. Hardt

Experimental evidence suggests that processing a model to equalize the error rates in different demographic groups is Pareto-optimal: varying the group-specific acceptance threshold yields the fairness-accuracy frontier.

**On the joint interaction
of models, data and features**
Y. Jiang et al. (2024)

According to the GDE (generalization disagreement equality), the (expected) agreement in a deep ensemble (an ensemble of randomly initialized networks, on unlabeled data) equals its (expected) test accuracy – but this only holds for *calibrated* ensembles.

The agreement can be modeled by a tensor

$$\Omega \in \{0, 1\}^{\#models \times \#samples \times \#features}$$

built as follows:

- Fit M classification models;
- For each of them, define “features” as the principal components of its penultimate layer;
- Cluster the features (greedy algorithm for k -partite matching) attempting to put in the same cluster principal components whose correlation is above some threshold;
- Match features to observations.

Experiments suggest that the agreement, computed from that tensor, is very close to the test accuracy, even for non-calibrated ensembles.

Assessing generalization via disagreement
Y. Jiang et al.

Given a well-calibrated ensemble of classification models (same model, trained with a stochastic algorithm and different random seeds), the disagreement rate (on unseen, unlabeled data) equals the test error (in expectation).

**Statistically optimal k -means clustering
via nonnegative low-rank
semidefinite programming**
Y. Zhuang et al.

The k -means problem

$$\text{Minimize}_{\beta_1, \dots, \beta_k \in \mathbb{R}^p} \sum_{i=1}^n \text{Min}_{k \in [1, k]} \|X_i - \beta_k\|_2^2$$

is NP-complete and often solved with heuristics or relaxations: Lloyd’s algorithm, spectral clustering, NMF, SDP. The problem can be reformulated as

$$\text{Maximize}_{G_1 \sqcup \dots \sqcup G_k = [1, n]} \sum_k \frac{1}{|G_k|} \sum_{i, j \in G_k} \langle X_i, X_j \rangle$$

or

$$\text{Minimize}_{\substack{H \in \{0, 1\}^{n \times k} \\ H\mathbf{1}_k = \mathbf{1}_n}} \langle A, HBH^\top \rangle$$

where

$$A = -X'X$$

$$B = \text{diag}(|G_1|^{-1}, \dots, |G_k|^{-1})$$

The SDP relaxation looks for $Z = HBH^\top$ instead:

$$\text{Minimize}_{Z \succeq 0} \langle A, Z \rangle \quad \text{st} \quad \text{tr } Z = k, \quad Z\mathbf{1}_n = \mathbf{1}_n, \quad Z \succeq 0.$$

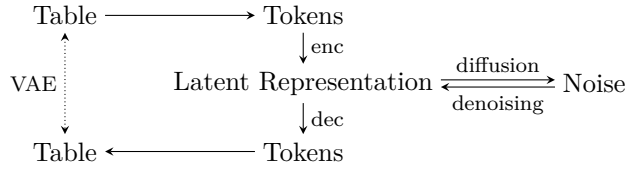
Further relax it by assuming Z is low-rank, $Z = UU'$ (and replacing $UU^\top \succeq 0$ with the stronger $U \succeq 0$)

$$\text{Minimize}_{U \in \mathbb{R}^{n \times r}} \langle A, UU^\top \rangle \quad \text{st} \quad \|U\|_F^2 = k, \quad UU^\top \mathbf{1}_n = \mathbf{1}_n, \quad U \succeq 0$$

(this keeps most of the theoretical properties of the SDP relaxation). The NMF relaxation did not have those equality constraints:

$$\text{Minimize}_{U \in \mathbb{R}^{n \times r}} \|A + UU^\top\|_F^2 \quad \text{st} \quad U \succeq 0.$$

**Mixed-type tabular data synthesis
with score-based diffusion in latent space**
H. Zhang et al.



**On the humanity of conversational AI:
evaluating the psychological portrayal of LLMs**
J.T. Huang et al.

Comparison of the results of psychological tests for humans and LLMs: LLMs lie more, and are more self-confident.

**What data benefits my classifier?
Enhancing model performance and
interpretability through influence-based
data selection**
A. Chhabra et al.

The *influence* of a sample (x_j, y_j) on some quantity of interest f (loss, fairness, robustness, etc.) is

$$I(x_j) = \nabla_\theta f(\hat{\theta})^\top H_\theta^{-1} \nabla_\theta \ell(x_j, y_j; \hat{\theta})$$

where

$$H_\theta = \frac{1}{n} \sum_i \nabla_\theta^2 \ell(x_i, y_i; \hat{\theta})$$

$$\hat{\theta} = \text{Argmin}_\theta \frac{1}{n} \sum_i \ell(x_i, y_i; \theta).$$

To interpret it, train a decision tree

$$(x_j, y_j) \mapsto I(x_j).$$

Use *hierarchical shrinkage* on the resulting tree.

***Hierarchical shrinkage:
improving the accuracy and
interpretability of tree-based methods***
A. Agarwal et al. (2022)

Consider a decision tree h :

q : input
 t_0 : root
 t_n : leaf containing q
 $t_n \subset t_{n-1} \subset \dots \subset t_0$: leaf-to-root path
 $\phi(t)$: number of sampled in node t
 $\xi(q, t)$: output for q in node t .

Its output

$$h(q) = \xi(q, t_0) + \sum_{j \geq 1} \xi(q, t_j) - \xi(q, t_{j-1})$$

can be shrunk to

$$\hat{h}(q) = \xi(q, t_0) + \sum_{j \geq 1} \frac{\xi(q, t_j) - \xi(q, t_{j-1})}{1 + \lambda / \phi(t_{j-1})}.$$

This alternative to pruning also applies to random forests: it often improves accuracy and simplifies the decision boundaries.

Code in `imodels`.

***Never train from scratch: fair comparison
of long-sequence models
requires data-driven priors***
I. Amos et al.

Do not train your models from a random initialization: pretrain them, on the downstream data (*self-pretraining*), with some denoising objective.

Flow matching on general geometries
R.T.Q. Chen and Y. Lipman

***Stochastic interpolants:
a unifying framework for flows and diffusions***
M.S. Albergo et al.

Given two probability distributions p_0 and p_1 , we want to progressively transform samples from p_0 into samples from p_1 (or the opposite).

Score-based diffusion models consider an OU process transforming p_0 into an (approximate) Gaussian p_0 , we sample x_1 from p_1 (a Gaussian) and use the backward SDE (its drift involves the score function $\nabla_{!x} \log p_t(x)$, which has to be learned) to get a corresponding x_0 . The path $(x_t)_t$ is stochastic, but one can use an ODE instead, and get a deterministic path.

Many paths give rise to the same intermediate distribution p_t : deterministic, with straight lines (optimal transport) or not (normalizing flows, in discrete “time”, or neural ODE, in continuous time), or stochastic.

A *stochastic interpolant* between p_0 and p_1 is a stochastic process x of the form

$$x_t = I(t, x_0, x_1) + \gamma(t)z, \quad t \in [0, 1]$$

where

$$\begin{aligned} I(0, x_0, x_1) &= x_0 \\ I(1, x_0, x_1) &= x_1 \\ \gamma(0) &= \gamma(1) = 1 \\ \gamma &\geq 0 \\ (x_0, x_1) &\sim \nu \\ \text{pr}_{1\#}\nu &= p_0, \text{ pr}_{2\#}\nu = p_1 \\ z &\sim N(0, I). \end{aligned}$$

For instance,

$$x_t = (1-t)x_0 + tx_1 + \sqrt{1t(1-t)}z$$

or (pure noise for $t = \frac{1}{2}$)

$$x_t = \cos^2(\pi t) \left(\mathbf{1}_{t < \frac{1}{2}} x_0 + \mathbf{1}_{t > \frac{1}{2}} x_1 \right) + \sqrt{2t(1-t)}z.$$

The density $(p_t)_t$ satisfies the *transport equation*

$$\partial_t p + \nabla \cdot (bp) = 0$$

where the *velocity* is

$$\begin{aligned} b(t, x) &= \mathbb{E}[\dot{x}_t | x_t = x] \\ b &= \underset{b}{\text{Argmin}} \mathbb{E}_{\substack{(x_0, x_1) \sim \nu \\ z \sim N(0, I) \\ t \sim \text{Unif}(0, 1)}} \left[\frac{1}{2} |b(t, x_t)|^2 = \dot{x}_t \cdot b(t, x_t) \right] \\ b &= v - \dot{\gamma} s \end{aligned}$$

Its *score* is

$$\begin{aligned} s &= \nabla_x \log p_t(x) = \gamma^{-1} \mathbb{E}[z | x_t = x] \\ &= \underset{x_0, x_1, z, t}{\text{Argmin}} \mathbb{E} \left[\frac{1}{2} |s(t, x_t)|^2 + \gamma(t)^{-1} z \cdot s(t, x_t) \right] \\ &= \underset{s}{\text{Argmin}} \mathbb{E}[|s|^2 + e \nabla \cdot s]. \end{aligned}$$

The *denoiser* is

$$\begin{aligned} \eta(t, x) &= \mathbb{E}[z | x_t = x] \\ \eta &= \underset{x_0, x_1, z, t}{\text{Argmin}} \mathbb{E} \left[\frac{1}{2} |\eta(t, x_t)|^2 = z \cdot \eta(t, x_t) \right] \end{aligned}$$

The *velocity field* is

$$\begin{aligned} v(t, x) &= \mathbb{E}[\partial_t I(t, x_0, x_1) | x_t = x] \\ v &= \underset{v}{\text{Argmin}} \mathbb{E} \left[\frac{1}{2} |v|^2 - \nabla_t Y \cdot v \right]. \end{aligned}$$

The forward and backward *Fokker-Plank equations* are

$$\begin{aligned} \partial_t p + \nabla \cdot (b_F p) &= \varepsilon \Delta p & p(0) &= p_0 \\ \partial_t p + \nabla \cdot (b_B p) &= -\varepsilon \Delta p & p(1) &= p_1 \end{aligned}$$

where ε is a non-negative (or positive semi-definite) function, and the forward and backward drifts are

$$\begin{aligned} b_F(t, x) &= b(t, x) + \varepsilon(t)s(t, x) \\ b_B(t, x) &= b(t, x) - \varepsilon(t)s(t, x). \end{aligned}$$

They are more robust than the transport equation to errors in the velocity and scores.

The following processes have the same law as the stochastic interpolant x_t .

$$\begin{aligned} \frac{d}{dt}X_t &= b(t, X_t) & X_0 &\sim p_0 \\ dX_t^F &= b_F(t, X_t^F)dt + \sqrt{2\varepsilon(t)}dW_t & X_0^F &\sim p_0 \\ dX_t^B &= b_B(t, X_t^B)dt + \sqrt{2\varepsilon(t)}dW_t^B & X_1^B &\sim p_1 \\ & & W_t^B &= W_{1-t} \end{aligned}$$

The density p_t can be computed from solutions of the ODE

$$\begin{aligned} p_t(x) &= \exp\left(-\int_0^t \nabla \cdot b(\tau, X_{t,\tau}(x))d\tau\right) p_0(X_{t,0}(x)) \\ &= \exp\left(\int_t^1 \nabla \cdot b(\tau, X_{t,\tau}(x))d\tau\right) p_1(X_{t,1}(x)) \end{aligned}$$

where

$$\begin{aligned} \frac{d}{dt}X_{st}(x) &= b(t, X_{st}(x)) \\ X_{ss}(s) &= x. \end{aligned}$$

For the SDEs:

$$\begin{aligned} p_1^F(x) &= \mathbb{E}\left[\exp\left(-\int_0^1 \nabla \cdot b_F(t, Y_t^B)dt\right) p_0(Y_0^B) \middle| Y_1^B = x\right] \\ p_1^B(x) &= \mathbb{E}\left[\exp\left(-\int_0^1 \nabla \cdot b_B(t, Y_t^F)dt\right) p_1(Y_1^F) \middle| Y_0^F = x\right] \end{aligned}$$

Other interpolants include:

– Diffusive interpolant

$$\begin{aligned} x_t &= I(t, x_0, x_1) + \sqrt{2a(t)}B_t \\ B: &\text{Brownian bridge} \end{aligned}$$

– One-sided interpolant

$$\begin{aligned} x_t &= \alpha(t)x_0 + J(t, x_1) \\ \alpha: &0 \rightsquigarrow 1 \\ J: &0 \rightsquigarrow x_1 \end{aligned}$$

– Mirror interpolant (cf denoisers)

$$\begin{aligned} x_t &= k(t, x_1) + \gamma(t)z \\ k: &x_1 \rightsquigarrow x_1 \end{aligned}$$

– Linear interpolant

$$\begin{aligned} x_t &= \alpha(t)x_0 + \beta(t)x_1 + \gamma(t)z \\ \alpha(0) &= \beta(1) = 1 \\ \alpha(1) &= \beta(0) = \gamma(0) = \gamma(1) = 0 \\ \forall t \quad &\alpha^2 + \beta^2 + \gamma^2 = 1 \end{aligned}$$

The *Schrödinger bridge* problem is

$$\begin{aligned} &\text{Find} && u, p \\ &\text{To minimize} && \int \int_{\substack{t \in [0,1] \\ x \in \mathbf{R}^n}} |u|^2 p \\ &\text{Such that} && \partial_t p + \nabla \cdot (up) = \varepsilon \Delta p \\ &&& p(0) = p_0, p(1) = p_1 \end{aligned}$$

Its solution is $p, \nabla \lambda$, where

$$\begin{aligned} \partial_t p + \nabla \cdot (\nabla \lambda p) &= \varepsilon \Delta p \\ \partial_t \lambda + \frac{1}{2} |\nabla \lambda|^2 &= \varepsilon \Delta \lambda. \end{aligned}$$

***How to avoid machine learning pitfalls:
a guide for academic researchers***
M.A. Jones

Flow matching for generative modeling
Y. Lipman et al.

A time-dependent vector field $v : [0, 1] \times \mathbf{R}^d \rightarrow \mathbf{R}^d$ defines a flow $\phi : [0, 1] \times \mathbf{R}^d \rightarrow \mathbf{R}^d$

$$\begin{aligned} \dot{\phi}_t(x) &= v_t(\phi_t(x)) \\ \phi_0(x) &= x \end{aligned}$$

and a probability density path

$$p_t = (\phi_t)_*(p_0)$$

or, equivalently (*continuity equation*)

$$\dot{p} + \text{div}(pv) = 0.$$

For each observation $x_1 \in \text{Data}$, the optimal transport from $N(0, I)$ to $N(x_1, \sigma_{\min}^2)$ defines the *conditional probability path*

$$x_t \sim N(tx_1, [1 - \sigma_{\min}]^2) = p_t(\cdot | x_1)$$

and the corresponding conditional vector field

$$u_t = \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}$$

(the deterministic probability flow of denoising diffusion models is more complex and can overshoot).

Find a vector field v_t (a deep neural network) v_t minimizing the *conditional flow matching* (CFM) objective

$$\mathbb{E}_{\substack{t \sim \text{Unif}(0,1) \\ x_1 \sim \text{Data} \\ x \sim p_t(\cdot | x_1)}} \|v_t(x) - u_t(x | x_1)\|^2$$

(easier to train than score matching).

**Improving and generalizing
flow-based generative models
with minibatch optimal transport**
A. Tong et al.

The flow-matching objective

$$\mathcal{L}(\theta) = \mathbb{E}_{\substack{t \sim \text{Unif}(0,1) \\ x_1 \sim \text{Data} \\ x \sim p_t(\cdot|x_0)}} \|v_\theta(t, x) - u_t(x|x_1)\|^2$$

where

$$p_t = N(tx_1, (t\sigma - t + 1)^2)$$

$$u_t(x|x_1) = \frac{x_1 - (1 - \sigma)x}{1 - (1 - \sigma)t}$$

can be generalized to

$$\mathbb{E}_{\substack{t \sim \text{Unif}(0,1) \\ x_0, x_1 \sim q \\ x \sim p_t(x_0, x_1)}} \|v_\theta(t, x) - u_t(x|x_0, x_1)\|^2$$

where

$$q(x_0, x_1) = q(x_0)q(x_1) \text{ (independent coupling)}$$

$$u_t(x|x_0, x_1) = x_1 - x_0$$

$$p_t(\cdot|x_0, x_1) = N(tx_1 + (1 - t)x_0, \sigma^2)$$

or q = optimal transport from q_0 to q_1 , or the entropy-regularized optimal transport.

$$q = \underset{\substack{(\text{pr}_0)_{\#} \pi = q_0 \\ (\text{pr}_1)_{\#} \pi = q_1}}{\text{Argmin}} \text{KL}(\pi \| \pi_{\text{ref}})$$

$$p_t = N(tx_1 + (1 - t)x_0, t(1 - t)\sigma^2)$$

$$u_t = \frac{1 - 2t}{2t(1 - t)} [x - (tx_1 + (1 - t)x_0)] + (x_1 - x_0)$$

Implementation in `torchcfm`.

**Minibatch optimal transport distances;
analysis and applications**
K. Fatras et al.

Computing the optimal distance on minibatches and averaging is efficient and biased (and it is no longer a distance); it can be debiased.

Poisson flow generative models
Y. Xu et al.

Given a probability distribution on \mathbf{R}^n , put electric charges in $[z = 0] \subset \mathbf{R}^n \times \mathbf{R}$ according to it, and consider the motion of a particle (of the same charge) starting at $(x, \varepsilon) \in \mathbf{R}^n \times \mathbf{R}$, $\varepsilon > 0$: after a while, those particles will be uniformly distributed on a hemisphere.

To generate data from this distribution, start with uniform data on a large hemisphere, and let it evolve along the reverse process.

Learn the Poisson field from data.

$\Delta\phi = -\rho$	Poisson equation
$\rho : \mathbf{R}^{n+1} \rightarrow \mathbf{R}$	Source function
$\phi : \mathbf{R}^{n+1} \rightarrow \mathbf{R}$	Potential function
$E = -\nabla\phi$	Poisson field
$\frac{\partial p}{\partial t} = -\nabla \cdot (pE)$	Gradient flow
$\phi(x) = \int G(x, y)\rho(y)dy$	
$G(x, y) \propto \ x - y\ ^{(n+1)-2}$	

**PFGM++: unlocking the potential
of physics-inspired generative models**
Y. Xu et al.

PFGM adds one dimension; instead, add D dimensions. The limit $D \rightarrow \infty$ corresponds to diffusion models, but smaller values give more robust models.

**Elucidating the design space
of diffusion-based generative models**
T. Karras et al.

Diffusion models consider a process in which we progressively add noise to a sample from the data distribution

$$x_0 \sim p_0 \text{ (data)}$$

$$x_{k+1} = x_k + \varepsilon_{k+1}, \quad \varepsilon_{k+1} \sim N(0, \sigma_{k+1}^2 I).$$

For $k \gg 1$, x_k is approximately Gaussian

$$x_k \sim p_k \approx N(0, \sigma^2 I).$$

Diffusion models try to reverse that process, starting with $x_k \sim N(0, \sigma^2 I)$, and trying to find $x_0 \sim p_0$.

Instead of looking at samples x_k , we can look at the corresponding distributions

$$p_{k+1} = p_k * N(0, \sigma_{k+1}^2 I).$$

In the continuous limit, we have a probability flow

$$x(0) \sim p_0$$

$$\dot{x} = -\dot{\sigma} \nabla_x \log p_t.$$

Deterministic diffusion models sample from p_0 as follows:

- Learn the score function $s = \nabla_x \log p_t$ from data;
- Sample from $x(1) \sim N(0, \sigma^2 I) \approx p_1$;
- Solve the ODE using $x(1)$ as final condition;
- x_0 is then distributed as p_0 .

The ODE $\dot{y} = f(t, y)$ can be solved with Euler's method

$$y_{i+1} = y_i + hf(t_i, y_i)$$

or *Heun's method* (aka improved Euler, trapezoidal)

$$\tilde{y}_{i+1} = y_i + hf(t_i, y_i)$$

$$y_{i+1} = y_1 + h \frac{f(t_i, y_1) + f(t_{i+1}, \tilde{y}_{i+1})}{2}.$$

If we keep the stochasticity, the ODE becomes an SDE

$$dx_{\pm} = -\dot{\sigma}\sigma s dt \pm \beta\sigma^2 s dt + \sqrt{2\beta}\sigma dW$$

where $\beta = \dot{\sigma}/\sigma$ (other choices are possible), *i.e.*

$$\begin{aligned} dx_+ &= \sqrt{2\beta}\sigma dW && \text{adding noise} \\ dx_- &= -2\dot{\sigma}\sigma s dt + \sqrt{2\beta}\sigma d\bar{W} && \text{denoising.} \end{aligned}$$

This is Anderson's *time reversal* formula

$$\begin{aligned} dX &= \mu dt + \sigma dW \\ d\bar{X} &= (\mu - \sigma^2 s) dt + \sigma d\bar{W} \\ s &= \nabla_x \log p \end{aligned}$$

(valid if $\sigma = \sigma_t$ does not depend on t). As in the deterministic case, *stochastic diffusion* models sample from p_0 :

- Learn the score function from data;
- Sample $x(1) \sim N(0, \sigma^2 I) \approx p_1$;
- Solve (*i.e.*, sample from) the backward SDE;
- $x(0)$ is then distributed as p_0 .

To solve (*i.e.*, sample from) an SDE $dX = \mu dt + \sigma dW$, Euler-Maruyama performs an ODE step and then adds noise;

$$\begin{aligned} \text{Euler} \quad \Delta X &= \mu \Delta t + \sigma \Delta W \\ \text{Milstein} \quad \Delta X &= \mu \Delta y + \sigma \Delta W + \frac{1}{2} \sigma \sigma' [(\Delta W)^2 - \Delta y] \\ \text{RK} \quad \hat{y}_n &= y_n + \mu(y_n) \Delta t + \sigma(y_n) (\Delta t)^{1/2} \\ y_{n+1} &= y_n + \mu(y_n) \Delta t + \sigma(y_n) \Delta W + \\ &\quad \frac{\sigma(\hat{y}_n) - \sigma(y_n)}{2} \frac{(\Delta W)^2 - \Delta t}{\sqrt{\Delta t}} \end{aligned}$$

reversing those operations (pay attention to the step size: it has to remain the same SDE).

$$\begin{aligned} \tilde{x}_{i+1} &= x_1 + \mu(x_i, t_i) h \\ x_{i+1} &= \tilde{x}_{i+1} + \sigma(x_i, t_i) (W_{i+1} - W_i) \end{aligned}$$

The score can be computed from a *denoiser*

$$\begin{aligned} D(\cdot, \sigma) &= \underset{D}{\text{Argmin}} \mathbb{E}_{y \sim \text{Data}} \mathbb{E}_{\varepsilon \sim N(0, \sigma^2 I)} \|D(y + \varepsilon, \sigma) - y\|^2 \\ \nabla_x \log p(x, \sigma) &= \frac{D(x, \sigma) - x}{\sigma^2} \end{aligned}$$

Do not learn D directly (the magnitudes of the inputs and outputs depend on σ) but F

$$D(x, \sigma) = x - \sigma F(x)$$

or, better

$$D(x, \sigma) = c_1(\sigma) + c_2(\sigma) F\left(\frac{\sigma \cdot \sigma_{\text{data}}}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}} c_3(\sigma) x, \frac{1}{\sqrt{\sigma^2 + \sigma_{\text{data}}^2}} c_4(\sigma) \log \sigma\right)$$

In the loss, use $1/c_2^2$ as weights.

Use data augmentation but, to prevent the augmentations from leaking into the generated images, use the augmentation parameters as conditioning inputs for F , and set them to zero during inference.

The unreasonable effectiveness of deep features as a perceptual metric R. Zhang et al.

The average distance between latent representations is more consistent with human evaluations than traditional metrics. Add a linear layer after the latent representation, or fine-tune the whole network, if you have data (LPIPS, learned perceptual image patch similarity).

Tackling decision processes with non-cummulative objectives using reinforcement learning M. Nägele et al.

Non-cummulative MDPs (NCMDP) maximize an arbitrary function of the rewards, rather than their discounted sum: Sharpe ratio, weakest link (maximize the minimum reward), optimization (find the state with the maximum reward). To apply classical RL algorithms, turn the NCMDP into an MDP:

- Replace the rewards with $f(r_{\leq t}) - f(r_{< t})$;
- Augment the state space to make the decision process Markov (sufficient statistics).

Text embeddings reveal (almost) as much as text J.X. Morris et al.

To invert vector embeddings of text, one could learn a mapping

$$\begin{cases} \text{embeddings} \longrightarrow \text{distributions on text} \\ e \longmapsto p(t|e). \end{cases}$$

Instead, train an error-correcting model $(e, \varepsilon) \longmapsto p(t|e, \varepsilon)$ and apply it iteratively

$$e \longmapsto t \longmapsto (e, \text{emb}(t) - e) \longmapsto t \longmapsto \dots$$

(4 times is enough).

Lag-Llama: towards foundation models for probabilistic time series forecasting K. Rasul et al.

Pretrain a transformer on 8000 time series (with stratified sampling), using lag features and time features, and Freq-Mix, Freq-Mask data augmentation, for univariate probabilistic forecasts (output the parameters of a Student T distribution). Also check: AutoArima, AutoETS, CrostonSBA, DynOptTheta, NPTS; AutoGluon (DeepAR, PatchTST, TFT); N-BEATS, Informer, Autoformer, ETSFormer; OneFitsAll.

MOMENT: A family of open time-series foundation models
M. Goswami et al.

The *time series pile* is a large collection of time series from various domains. Train a simple transformer (3 sizes) for masked prediction (divide the time series into patches, mask some of them, project to a d -dimensional embedding, apply a transformer, and reconstruct the patches); use for forecasting, prediction, anomaly detection, imputation.

Non-transformer approaches may work better: ARIMA (short-horizon forecasting), N-BEATS (long-horizon), k -NN (anomaly detection). Also check: Time-LLM, GPT4TS (OneFitsAll), TimesNet, PatchTST, FedFormer, DLinear, Stationary, LightTS, SeasonalNaive, AnomalyTransformer

One Fits All: power general time series analysis by pretrained LM
T. Zhou et al.

LLMs are of the form

input → encoder → transformers → decoder → output.

Freeze the transformers (the “intelligence” of the model) and retrain just the encoder (patch embedding) and decoder, for time series tasks (imputation, classification, anomaly detection, long-term/short-term/few-shot/zero-shot forecasting).

A time series is worth 64 words: long-term forecasting with transformers
Y. Nie et al.

PatchTST (time series transformer):

- Does not use pointwise attention, but segments the time series into patches;
- Independently processes the components of a multivariate time series, with shared weights for the embedding and transformers.

Informer: beyond efficient transformer for long sequence time series forecasting
H. Zhou et al.

The sparsity of a query q_i is

$$\text{KL}(\text{Unif} \| p(k_j | q_i));$$

up to a constant

$$\log \sum_j \exp \frac{q_i k_j^\top}{\sqrt{d}} - \text{Mean}_j \frac{q_i k_j^\top}{\sqrt{d}}.$$

ProbSparse attention is

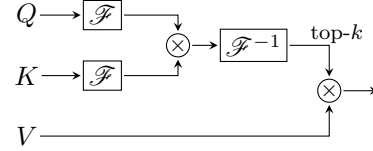
$$(Q, K, V) \mapsto \text{Softmax} \left(\frac{\bar{Q} K^\top}{\sqrt{d}} \right) V$$

where \bar{Q} only contains the top- u entries of Q (the others are uninformative).

Self-attention distillation reduces the time dimension (1-dimensional convolution, width=3, maxpooling, stride=2).

Autoformer: decomposition transformers with autocorrelation for long-term series forecasting
H. Wu et al.

Autocorrelation self-attention uses a convolution between keys and values instead of a product.



FEDFormer: frequency enhanced decomposed transformer for long-term series forecasting
T. Zhou et al.

Use a seasonal-trend decomposition, and apply the transformer in the frequency domain.

ITransformer: inverted transformers are effective for time series forecasting
Y. Liu et al.

(For multivariate time series), instead of attention between timestamps, use attention between time series.

Anomaly transformer: time series anomaly detection with association discrepancy
J. Xu et al.

Anomalies, in a time series, may be similar to their immediate neighbourhood, but not the whole series. The *anomaly transformer* measures this by the discrepancy (symmetrized KL divergence) between the “prior association”

$$P_i = \phi \left(-\frac{|j-i|}{\sigma_i} \right)_{1 \leq j \leq N}$$

and the “series association”

$$S_i = \text{Softmax} \left(\frac{Q_i K^\top}{\sqrt{d}} \right).$$

The prior association is trained to minimize discrepancy. The series association is trained to maximize discrepancy and reconstruct the input.

Mamba: linear time sequence modeling with selective state spaces
A. Gu and T. Dao

The S4 model is a continuous model

$$\begin{aligned} \dot{h} &= Ah + Bx \\ y &= Ch, \end{aligned}$$

discretized

$$\begin{aligned} h_t &= \bar{A}h_{t-1} + \bar{B}x_t \\ y_t &= Ch_t \end{aligned} \quad \begin{array}{c} y_t \\ \uparrow C \\ h_t \\ \xleftarrow{B} h_{t-1} \\ \uparrow A \\ x_t \end{array}$$

The discretization can be naive

$$\begin{aligned} \bar{A} &= I + hA \\ \bar{B} &= hB \end{aligned}$$

or not

$$\begin{aligned} \bar{A} &= \exp(\Delta A) \\ \bar{B} &= (\Delta A)^{-1}(\bar{A} - I)(\Delta B). \end{aligned}$$

The matrix A is structured, e.g., diagonal. The state space model (SSM) is applied independently to each channel. Since the model's dynamics are constant over time (linear time invariant (LTI) model), it can be computed efficiently.

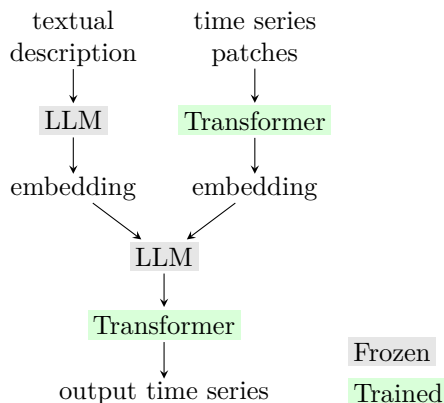
In the S6 model, B , C , Δ depend on the input x (linearly, with a softplus for Δ). It can still be implemented efficiently.

A decoder-only foundation model for time series forecasting **A. Das et al.**

Yet another time series foundational model, which breaks down the time series into patches (with larger patches in the output than in the input, for long-term forecasts).

Time-LLM: time series forecasting by reprogramming large language models **M. Jin et al.**

Use a pretrained LLM for time series forecasting: as inputs, provide an embedding (pre-trained, frozen) of a textual description of the time series (metadata in the prompt) and a sequence of patch embeddings (trained transformer); process with a pretrained, frozen LLM; and decode the output into a time series (trained transformer).



Learning programs by learning from failures **A. Cropper and R. Morel**

Popper is an inductive logic programming (ILP) system (learning from failures, LFF):

- Start with a set of hypotheses;
- Pick one of them, check if it fits the data;
- Prune the set of hypotheses if it does not (remove its generalizations if the hypothesis entails a negative example (too general), remove its specializations if it does not entail all the positive examples (too specific)).

Also check: Aleph, Metagol (Prolog), ILASP3 (answer set programming), ∂ ILP (neural nets).

Direct least squares fitting of ellipses **A. Fitzgibbon et al. (1999)**

To fit an ellipse (not an arbitrary conic) to a cloud of points

$$\begin{aligned} \text{Find} \quad & \underline{a} = (a \ b \ c \ d \ e \ f)' \\ \text{To minimize} \quad & \sum_i F(\underline{a} \cdot \underline{x}_i)^2 \\ \text{Subject to} \quad & b^2 - 4ac = -1 \\ \text{Where} \quad & \underline{x} = (x^2 \ xy \ y^2 \ x \ y \ 1)' \end{aligned}$$

(the constraint can be written $a'Ca = 1$).

Using Lagrange multipliers, this reduces to a generalized eigenvalue problem

$$\begin{aligned} D'Da &= \lambda CA \\ a'Ca &= 1 \end{aligned}$$

where the design matrix is $D = [\underline{x}_1, \dots, \underline{x}_n]'$. The solution is given by the unique positive eigenvalue.

Topological time series analysis **J.A. Perea**

The *maximum persistence* (length of the longest interval in the 1-dimensional barcode) is a useful feature, e.g., for periodicity quantification (gene expression, video) and regime (change point) detection.

Sliding windows and persistence: an application of topological methods to signal analysis **J.A. Perea and J. Harer (2013)**

Maximum persistence can be used to quantify periodicity (deciding whether a time series is periodic). Also check: `jtk_cycle`, Lomb-Scargle.

Jtk_cycle: an efficient non-parametric algorithm for detecting rhythmic components in genome-scale datasets **M.E. Hughes et al. (2010)**

To test if a signal is periodic:

$$\underset{\phi, T}{\text{Argmax}} \tau(x[i], s_{\phi, T}[i])$$

where x is the data, $i = \text{argsort}(x)$, τ is Kendall's tau, and $s_{\phi,T}$ is a signal with phase ϕ and period T – but the objective function oscillates a lot. R implementation in `MetaCycle`.

MetaCycle: an integrated R package to evaluate periodicity in large scale data
G. Wu et al.

Universal differential equations for scientific machine learning
C. Rackaukas et al. (2021)

UDEs model physical systems as $x'' = f_{\theta}(x) + \varepsilon(x)$, where

- f_{θ} is a known function corresponding to a known approximation of the phenomenon (or 0);
- θ are unknown parameters (mass of an object, charge of a particle, etc.);
- ε is a free-form unknown function (e.g., a neural net);
- The ODE can be replaced with a PDE.

Down with determinants
S. Axler (1995)

Most of linear algebra (eigenvalues, minimal polynomial, characteristic polynomial, Jordan form, etc.) can be done without determinants – they are still needed for the change-of-variable formula in multiple integrals, but they can be defined as the product of the eigenvalues, with multiplicity. More details in his book, *Linear algebra done right*.

A survey of the Schrödinger problem and some of its connections with optimal transport
C. Léonard (2014)

The *dynamic Schrödinger problem* (Schrödinger bridge) is

$$\text{Minimize}_P \text{KL}(P\|R) \text{ such that } P_0 = \mu_0, P_1 = \mu_1$$

where

- P is a probability measure on $\mathcal{C}^0([0, 1], \mathbf{R}^n)$;
- R is the law of Brownian motion on \mathbf{R}^n (unbounded measure);
- μ_0, μ_1 are probability measures on \mathbf{R}^n (prescribed marginals).

The *static Schrödinger problem* is

$$\text{Minimize}_{\pi} \text{KL}(\pi\|R_{01}) \text{ such that } \pi_0 = \mu_0, \pi_1 = \mu_1$$

where

- R_{01} is the joint law of the initial and final positions of R ;
- π is a probability measure on $(\mathbf{R}^n)^2$.

The two are linked by

$$P(\cdot) = \int_{(\mathbf{R}^n)^2} R^{xy}(\cdot) \pi(dx dy)$$

where R^{xy} is the Brownian bridge from x to y

$$R^{xy}(\cdot) = R(\cdot | X_0 = x, X_1 = y)$$

(the Schrödinger bridge is a mixture of Brownian bridges, governed by π).

These are eerily similar to the (Monge-Kantorovich) dynamical and static optimal transport problems.

An introduction to Otto's calculus
L. Ambrosio et al. (2021)

The Wasserstein distance on $\mathcal{P}_2^a(\mathbf{R}^2)$, the space of absolutely continuous measures $\mu = \rho dx$, comes from the scalar product on $T_{\rho}\mathcal{P}_2^a(\mathbf{R}^n)$

$$\langle s, s' \rangle = \int \langle \nabla \phi, \nabla \phi' \rangle \rho dx$$

where

- $s, s' \in T_{\rho}\mathcal{P}_2^a(\mathbf{R}^n)$, i.e., they are functions with zero mean, $\int s \rho dx = 0$;
- $-\text{div}[(\nabla \phi)\rho] = s$;
- $-\text{div}[(\nabla \phi')\rho] = s'$.

The gradient flow ρ_t for the vector field $\nabla \phi_t$ is the solution of the continuity equation

$$\frac{d}{dt}\rho_t + \text{div}[(\nabla \phi_t)\rho_t] = 0.$$

Comment: causal inference competitions: where should we aim?
E. Karavani et al.

Because of the “fundamental problem of causal inference” (we cannot observe counterfactuals), we have to limit ourselves to simulation studies. But they rely on strong/arbitrary assumptions, which may not hold in real setups: vary the parameters of the DGP, and score them separately (only aggregate the scores if you need to select a single winner). It is possible to use real observational data, when there is a soon-to-be published experimental trial.

Adjustment identification distance: a gadjid for causal structure learning
L. Henckel et al.

To define a causal distance between two DAGs, g_{true} and g_{guess}

- For each pair of nodes (TmY), compute a valid adjustment set in g_{guess} ;
- Check if it is a valid adjustment set in g_{true} ;
- Count the discrepancies.

There are many possible choices for the valid adjustment set:

- The parents of T (this gives the SID, structural intervention “distance” – note that $d(g_{\text{true}}, g_{\text{guess}}) = 0$ iff $g_{\text{true}} \subset g_{\text{guess}}$);
- All the nodes before T in a topological order;
- The “optimal adjustment set”.

This can be generalized to other (non-adjustment-based) identification strategies, provided you can check if it is valid.

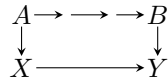
Implementation (Rust/Python) for DAGs and CPDAGs in `gadjid`.

**Graphical criteria
for efficient total effect estimation
via adjustment in causal linear models
L. Henkel et al. (2020)**

Parent adjustment is not optimal, in terms of asymptotic variance. Instead, use $\text{pa}(\text{cn}) \setminus \text{forb}$, where

- cn are the “causal nodes” of $X \rightarrow Y$, *i.e.*, nodes on a directed path $X \rightarrow \dots \rightarrow Y$, excluding X but including Y (mediators);
- pa are the parents;
- $\text{forb} = \text{de}(\text{cn}) \cup \{X\}$: forbidden nodes;
- $\text{dec}(\text{cn})$ are the descendants of cn , including cn .

The adjustment set prefers variables explaining less of X and more of Y . For instance, if the DAG is



do not condition on A , but on B .

**Measuring causality with the
variability of the largest eigenvalue
A.R. Dominguez and O.H. Yadav (2024)**

$\lambda_1/(\lambda_1 + \lambda_2)$ where $\{\lambda_1, \lambda_2\}$ is the spectrum of $\text{Var}[Y_t, X_{t-k}]$.

**Efficient causal graph discovery
using large language models
T. Jiralerspong et al.**

To learn causality from metadata, using an LLM, do not feed the pairs of variables one by one (“which is more likely? 1. $A \rightarrow B$ 2. $B \rightarrow A$ 3. $A \leftarrow \cdot \rightarrow B$ 4. $A \perp\!\!\!\perp B$) but progressively

- Which are the variables not caused by any other?
- Which are the variables caused directly by X ? (Give all the conclusions so far to the LLM; you can also provide more information, *e.g.*, $\text{Cor}(X, \cdot)$.)
- Continue, depth-first, until you have exhausted all the variables.

**A survey on causal discovery:
theory and practice
A. Zanga and F. Stella (2023)**

Conditional independence relations cannot distinguish between all DAGs: they can only recover the *Markov equivalence class* (MEC) of the causal model, which can be described by its CPDAG (completed partial directed acyclic graph), a PDAG where

- All undirected edges are reversible (you can choose either direction, that will not change the MEC);

- All directed edges are compelled (if you flip them, the graph moves to another MEC).

If we want to account for missing confounders, things get more complicated, and the graphs (partial ancestral graphs, PAG) have 4 types of edges:

- $X \rightarrow Y$ cause
- $X \leftrightarrow Y$ unobserved confounder
- $X \circ \rightarrow Y$ cause or missing confounder
- $X \circ \dashrightarrow Y$ any of the above

Allowing for cycles (feedback) complicates things further (σ -separation and directed mixed graphs).

Here are some causal discovery algorithms.

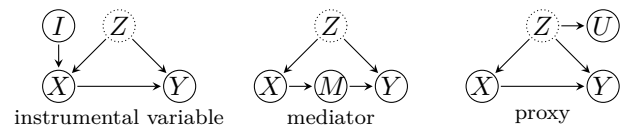
- The *PC algorithm* starts with a complete graph, uses conditional independence tests to remove edges and identify colliders, and finally orients some of the remaining edges, assuming we have found all the colliders.
- The *FCI algorithm* generalizes this to account for unobserved confounders.
- The *GES algorithm* (greedy equivalence search) adds the edges one by one, to increase a score (BIC, AIC, BDeu, BDs).
- *FGES* is an efficient implementation of GES, avoiding recomputations.
- Greedy FCI (*GFCI*), is FCI on the GES skeleton.
- *LinGAM* assumes the SCM is linear with non-Gaussian independent noise $x = Ax + \varepsilon$ (with A triangular), *i.e.*, $\varepsilon = (I - A)x$, and uses ICA to find the linear transformation of x whose coordinates maximize non-Gaussianity.
- *NOTEARS* looks for a matrix A such that $x \approx Ax$, with the constraint that it be acyclic – this can be formulated as $\text{tr} \exp |A| = n$, where A is the elementwise absolute value and n the number of variables.
- Interventional algorithms include GIES, IGSP, FCI-JCI, Ψ -FCI, DCDI, etc.

**Methods and tools
for causal discovery and causal inference
A.R. Nogueira et al. (2022)**

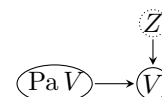
Another causal discovery review paper.

**Stochastic causal programming
for bounding treatment effects
K. Padh et al.**

In presence of unobserved confounders, auxiliary variables can help identify causal effects.



Stochastic causal programming (SCP) does not model the distribution of the unobserved confounders but, for each variable V



the distribution over function $f_{z,V}$

$$V = f_{z,V}(\text{Pa } V)$$

with basis functions, and constraints (smoothness, maximum number of inflection points, etc.). We can then compute the maximum and minimum of $E[Y \mid \text{do}(X = x)]$ over all plausible models.

A versatile causal discovery framework to allow causally related hidden variables
X. Dong et al.

D-separation can be inferred by conditional independence tests. *T-separation* can be inferred by the rank of the (cross)covariance matrix

$$\text{rank } \Sigma_{A,B} = \text{Min}\{|C_A| + |C_B| : A \perp\!\!\!\perp_t B \mid C_A, C_B\}.$$

It subsumes d-separation, and is more informative in presence of latent variables.

$$A \perp\!\!\!\perp_d B \mid C \text{ iff } \text{rank } \Sigma_{A \cup C, B \cup C} = |C|$$

A *trek* $X \xrightarrow{(P_1, P_2)} Y$ is a pair of paths with the same source, and X and Y as respective sinks. A and B are *t-separated* given (C_A, C_B) if, for every trek (P_1, P_2) from a node in A to a node in B , either $P_1 \cap C_A \neq \emptyset$ or $P_2 \cap C_B \neq \emptyset$.

Causal inference for time series analysis: problems, methods and evaluation
R. Moraffah et al.

Causal discovery for time series often relies on (generalizations of) Granger causality, the PC algorithm (PCMCI) or structural equation models (LiNGAM on VAR residuals).

Review of causal discovery methods based on graphical models
C. Glymour et al.

Clear explanation of the PC, FCI and LiNGAM algorithms.

Introduction to the foundations of causal discovery
F. Eberhardt (2016)

Use a MaxSAT solver to combine background knowledge with the output of a causal discovery algorithm.

A random variable (X_1, \dots, X_n) is *Markov* wrt a graph G if $\forall X \ X \perp\!\!\!\perp \text{NonDesc}(X) \mid \text{Pa}(X)$; this implies

$$X \perp\!\!\!\perp_d Y \mid C \implies X \perp\!\!\!\perp Y \mid C.$$

It is *faithful* to the graph if the converse holds:

$$X \perp\!\!\!\perp Y \mid C \implies X \perp\!\!\!\perp_d Y \mid C$$

(i.e., the graph has no extra edges, and there are no cancellations).

Causal discovery on high-dimensional data
Z. Hao et al. (2014)

To find a causal graph in high dimension:

- For each variable y , find its potential neighbours by using the max relevance min redundancy criterion

$$\text{Maximize}_S \sum_{x \in S} I(y; x) - \frac{1}{|X|} \sum_{x, x' \in S} I(x; x');$$

greedily:

$$x_{k+1} = \text{Argmax}_{x \in X \setminus S_k} I(y; x) - \frac{1}{k} \sum_{x' \in S_k} I(x; x')$$

$$S_{k+1} = S_k \cup \{x_{k+1}\}.$$

- Use conditional independence tests (χ^2 , G or kernel-based) to prune S and only keep direct causes and consequences of y ;
- Orient the edges with entropy-based IGCI: if x is more random than y , i.e., if $H(x) > H(y)$, then choose the direction $x \rightarrow y$.

A causal feature selection algorithm for stock prediction modeling
Z. Zhang et al. (2013)

Replace feature selection (lasso, etc.) with *causal feature selection*:

- Greedily add variables x with large $I(x; y)$, unless they are independent of x conditionally on the variables already selected;
- Identify the causes of y by noticing that y is a collider for them.

CausalTime: realistically generated time series for benchmarking of causal discovery
Y. Cgeng et al.

- Fit a non-linear VAR (NAR) (with noise modeled with a normalizing flow);
- Define the parents of a variable X_i as the variables with the largest feature importance (DeepShap);
- Split the NAR into 3 terms: causal (from the identified parents), residual and noise

$$x_i = f_i(x \odot H) + [f_i(x) - f_i(x \odot H)] + \varepsilon_i$$

where H is the adjacency matrix of the discovered parents;

- Use the model to generate new data, of dimension $2N$, keeping (separately) both the causal and residual terms, with adjacency matrix

$$\begin{bmatrix} H & \mathbf{1} \\ I & 0 \end{bmatrix}.$$

Beware of the simulated DAG! Causal discovery benchmarks may be easy to game
A.G. Reisach et al.

Sorting the variables by increasing variance can recover a topological order of the causal graphs; use the lasso (regressing each variable on its potential parents) to recover the graph.

If the data has been normalized, correlations tend to increase (decrease) in the causal (anti-causal) direction:

$$i \rightarrow j \rightarrow k \quad \text{Cor}(X_i, X_j) \leq \text{Cor}(X_j, X_k).$$

Approximate kernel-based conditional independence tests for fast nonparametric causal discovery
E.V. Strobl et al.

The *kernel conditional independence test* (KCIT) can be approximated with random features (randomized conditional independence test, RCIT):

$$\begin{aligned} H_0 : X \perp\!\!\!\perp Y \mid Z \\ \ddot{X} &= (X, Z) \\ \ddot{A} &= [f_1(\ddot{X}), \dots, f_m(\ddot{X})] \\ \ddot{B} &= [h_1(Y), \dots, h_q(Y)] \\ \ddot{C} &= [g_1(Z), \dots, g_d(Z)] \\ g_1(Z) &= \sqrt{2} \cos(W^\top Z + B) \\ W &\sim N(0, I) \\ B &\sim \text{Unif}(0, 2\pi) \\ \text{idem for the } f_i\text{'s and } h_i\text{'s} \\ \hat{\Sigma}_{\ddot{A}B \cdot C} &= \hat{\Sigma}_{\ddot{A}B} - \hat{\Sigma}_{AC}(\hat{\Sigma}_{CC} + \gamma I)^{-1} \hat{\Sigma}_{CB} \\ &= \text{Cov}[\text{res}(\ddot{A} \sim C), \text{res}(B \sim C)] \end{aligned}$$

(using ridge regression).

The *randomized conditional correlation test* (RCoT) uses X instead of \ddot{X} .

$$\text{test statistic} = n \left\| \hat{\Sigma}_{\ddot{A}B \cdot C} \right\|_F^2$$

(the distribution of the test statistic under H_0 is complicated: linear combination of χ^2 variables, with weights given by the eigenvalues of some covariance matrix).

Conditional independence testing based on a nearest neighbour estimator of conditional mutual information
J. Runge

Conditional mutual information

$$I(X; Y \mid Z) = H(X, Z) + H(Y, Z) - H(Z) - H(X, Y, Z)$$

can be estimated with the KL (Kozachenko-Leonenko) entropy estimator. To estimate its (finite-sample) distribution under

$$H_0 : X \perp\!\!\!\perp Y \mid Z$$

use random permutations, taking care to only destroy the relation between X and Y , while preserving that between X and Z .

Python implementation in `tigramite`.

Better simulations for validating causal discovery with the DAG-adaptation of the onion method
B. Andrews and E. Kummerfeld

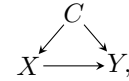
The onion method samples from the space of correlation matrices by building correlation matrices one dimension at a time.

$$R_{i+1} = \begin{pmatrix} R_1 & r_{i+1} \\ r'_{i+1} & 1 \end{pmatrix} \quad 1 - r'_{i+1} R_i^{-1} r_{i+1} < 1$$

It can be modified to sample from correlation matrices compatible with a given DAG.

Causal discovery from heterogeneous/nonstationary data
B. Huang et al.

CD-NOD is a PC-like causal discovery algorithm, which deals with heterogeneous and/or nonstationary data by adding a “domain” and/or “time” variable; variables connected to them have changing causal mechanisms. Assuming that the distribution shifts are due to a small number of variables also helps identify some edge directions: in the situation



if we see changes in $P(X|Y)$ and $P(Y)$ and $(P(X) \text{ xor } P(Y|X))$, we conclude that $X \rightarrow Y$.

Amortized causal discovery: learning to infer causal graphs from time series data
S. Löwe et al.

Neurons (in different patients s) have the same dynamics, but different connectivity (causal graphs) G_s

$$\underline{x}_s^{t+1} = g(\underline{x}_s^{\leq t}, G_s).$$

Model them, jointly, as

$$\begin{aligned} \hat{G}_s &= f_\phi(\underline{x}_s) \\ \underline{x}_s^{t+1} &\approx f_\theta(\underline{x}_s^{\leq t}, \hat{G}_s). \end{aligned}$$

Baselines:

- Linear Granger causality;
- Neural Granger causality, with MLP, LSTM, or eSRU (economy statistical recurrent unit);
- MPIR (minimum predictive information regularization);
- Transfer entropy;
- Mutual information.

Data: particles; phase-coupled oscillators; simulated fMRI (netsim).

Neural Granger causality

A. Tank et al.

Nonlinear Granger causality, with a sparse-input MLP or an LSTM.

Economy statistical recurrent units for inferring nonlinear Granger causality

S. Khanna and V.Y.F. Tan

The statistical recurrent unit (SRU) is yet another RNN/GRU/LSTM variant, with feedback. Use it, componentwise, for nonlinear Granger causality testing: the causal relations can be extracted from the model parameters. Reduce the number of parameters by reducing the dimension of (some of) the latent space(s) with sketching (fixed random projections), and add sparsifying penalties.

Discovering nonlinear relations with minimum predictive information regularization

T. Wu et al.

To test for Granger causality with neural nets, there is no need to try the variables one by one: allow each $X_{j,t-1}$ to have learnable corruption $\tilde{X}_{j,t-1}$, as much as possible (measure it with the mutual information between $\tilde{X}_{j,t-1}$ and $X_{j,t-1}$) while maintaining good prediction of X_{it} .

Causal structure learning supervised by large language model

T. Ban et al.

Use some (data-based) causal learning algorithm; then ask an LLM if the edges are correct, and add a constraint to remove the incorrect edges; iterate until convergence.

The DeCAMFounder: nonlinear causal discovery in the presence of hidden variables

R. Agrawal et al.

To estimate a causal graph in presence of *pervasive confounding* (unobserved confounders affecting many observed variables), use PCA to recover (sufficient statistics) of the unobserved variables.

ABIDES-Economist: agent-based simulation of economic systems with learning agents

K. Dwarakanath et al.

Multi-agent system, with households (labour, consumption, saving), firms (wages, prices, production, inventory), central bank (interest rate) and government (tax rate, tax credits).

RiskMiner: discovering formulaic alphas via risk-seeking Monte Carlo tree search

T. Ren et al.

Alpha mining, not with a genetic algorithm (gplearn), but with reinforcement learning (MCTS), with

- $\text{Cor}(\alpha, \text{returns}) - \text{Mean}_{\alpha'} \text{Cor}(\alpha, \alpha')$ as intermediate rewards, to ensure diversity;
- $\text{Cor}(\alpha, \text{returns}) = \text{IC}$ as final reward.

Do not optimize the expected cumulated reward, but some quantile of it.

DiffFormer: a diffusion transformer on stock factor augmentation

Y. Gao et al.

Diffusion models (DDPM) progressively add noise to their input

$$x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \varepsilon_t, \quad \varepsilon_t \sim N(0, I),$$

equivalently,

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \varepsilon, \quad \varepsilon \sim (, I),$$

and try to reverse the process

$$\text{Minimize}_{\theta} \mathbb{E}_{\substack{x_0 \sim \text{data} \\ \varepsilon \sim N(0, I) \\ t \sim \text{Uniform}}} \|\varepsilon - \varepsilon_{\theta}(x_t, t)\|^2.$$

Do not denoise pure noise, but noisy inputs (diffusion-based augmentation).

ABCs (and Ds) of understanding VARs

J. Fernández-Villaverde et al.

In a state space model

$$x_{t+1} = Ax_t + Bw_{t+1}$$

$$y_{t+1} = Cx_t + Dw_{t+1}$$

where x is hidden and w are stocks, when can we recover the shocks w_{t+1} by comparing y_{t+1} and $\mathbb{E}_t[y_t]$?

$$\text{DSGE} \rightarrow \text{SSM} \rightarrow \text{VAR}$$

DSGE models

A. Mikusheva (2007)

A DSGE model considers agents (firms, households, banks, state, etc.), each maximizing its utility, subject to a few balance equations. Writing the first order equations and linearizing everything gives a linear SSM, amenable to the Kalman filter (or a particle filter, if you do not linearize).

Properties of the entropic risk measure EVaR in relation to selected distributions

Y. Mishura et al.

Explicit computation of the entropic value at risk

$$\text{EVaR}_{\alpha} X = \inf_{t > 0} \frac{1}{t} \log \frac{m_X(t)}{1 - \alpha}$$
$$m_X(t) = \mathbb{E}[e^{tX}] \quad \text{moment generating function}$$

for a few distributions (using the Lambert W function, $W(x)e^{W(x)} = x$).

Plutos: towards interpretable stock movement prediction with financial large language model
H. Tong et al.

Have several models (technical analysis, sentiment, financial ratios, etc.) output their forecasts as text, with an explanation, and use another LLM to aggregate them, with an explanation.

Optimal text-based time series indices
D. Ardia and K. Bluteau (2024)

To convert text (news) into macroeconomic time series, one typically uses some (trained) sentiment indicator, computed from heuristically selected documents (based on the presence of keywords) – instead, the selection process should be part of the optimization process.

Application to EPU (economic policy uncertainty).

Modeling financial time series with generative adversarial networks
S. Takahashi et al. (2019)

Stylized facts of financial time series include:

- Linear unpredictability (no correlation);
- Fat tails, $3 < \alpha < 5$;
- Volatility clustering $\text{Cor}(|r_t|, |r_{t+k}|) \propto k^{-\beta}$;
- Leverage: $\text{Cor}(r_t, \sigma_{t+k}) < 0$;
- Coarse-fine volatility correlation

$$\begin{aligned} \text{coarse}_t &= \left| \sum_{1 \leq i \leq \tau} r_{t-i} \right| \\ \text{fine}_t &= \sum |r_{t-i}| \\ \rho(k) &= \text{Cor}(\text{coarse}_{t+k}, \text{fine}_t) \\ \Delta\rho(k) &= \rho(k) - \rho(-k) < 0 \end{aligned}$$

- Gain-loss asymmetry: the waiting time for a +10% price change is larger than that for a -10% change (prices drop faster than they rise).

Macroeconomic regimes
L. Baele et al. (2014)

Model inflation, output and interest rate as

$$\begin{aligned} \pi_t &= \delta \hat{\pi}_t + (1 - \delta)\pi_{t-1} + \lambda y_t + \text{noise} \\ y_t &= \mu \hat{y}_t + (1 - \mu)y_{t-1} - \phi(i_t - \hat{\pi}_t) + \text{noise} \\ i_t &= p \hat{i}_t + (1 - \rho)(\beta \hat{\pi}_t + \gamma y_t) + \text{noise} \end{aligned}$$

with regime switching for (β, γ) , $\text{Var } \varepsilon_\pi$, $\text{Var } \varepsilon_y$, $\text{Var } \varepsilon_i$.

Modeling the term structure of interest rates: an introduction
M. Fisher (2003)

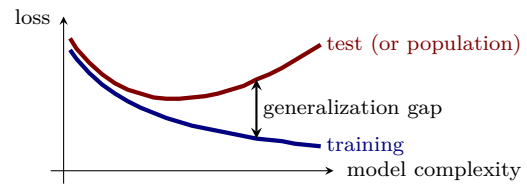
Pricing the term structure with linear regressions
T. Adrian et al. (2013)

Alternative risk premia timing: a point-in-time macro, sentiment, valuation analysis
O. Blin et al. (2018)

List of (multi-asset) long-short strategies.

Probabilistic machine learning
K.P. Murphy (2022)

1. A machine learning problem combines a task, training data, and a performance measure. *Epistemic* uncertainty (model uncertainty) reflects our ignorance of the model. *Aleatoric* uncertainty (data uncertainty) reflects the noise in the model.



Unsupervised learning can be seen as (interpretable) data compression.

2. Summary statistics have limitations (Anscombe quartet, datasaurus dozen).

In medicine, to compute $P[\text{infected}|\text{test}]$ with Bayes's rule, we need sensitivity, specificity and prevalence.

$$\frac{\text{sensitivity} \times \text{prevalence}}{\text{sensitivity} \times \text{prevalence} + (1 - \text{specificity}) \times (1 - \text{prevalence})}$$

$$y \sim \text{Bernoulli}(\sigma(w'x + b)) \quad \text{logistic regression}$$

$$y \sim \text{Cat}(\text{softmax}(Wx + b)) \quad \text{multinomial logistic regression}$$

The *log-sum-exp trick* is

$$\log \sum e^{a_i} = m + \log \sum e^{a_i - m};$$

with $m = \text{Max } a_i$, the $a_i - m$ are negative, and the exponential does not overflow.

Common distributions include Gaussian, Student, Laplace, Cauchy, beta, gamma, exponential, chi squared, inverse gamma.

$$\text{Exp}(\lambda) = \text{Gamma}(1, \lambda)$$

$$\chi^2(\nu) = \text{Gamma}\left(\frac{\nu}{2}, \frac{1}{2}\right)$$

If $X \sim p_X$ and $Y = f(X)$, then $Y \sim p_Y$, where the change-of-variable formula is

$$p_Y(y) = p_X(f^{-1}(y)) \left| \det \frac{df^{-1}(y)}{dy} \right|.$$

If $X \perp\!\!\!\perp Y$, then $p_{X+Y} = p_X * p_Y$.

3. *Simpson's paradox* says that a statistical trend that appears in different groups can disappear or reverse sign when those groups are combined.

The *conditional Gaussian distribution* is

$$\begin{aligned} \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} &\sim N\left(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right) \\ Y_1 | Y_2 = y_2 &\sim N(\mu_{1|2}, \Sigma_{1|2}) \\ \mu_{1|2} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(y_2 - \mu_2) \\ \Sigma_{1|2} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21} = \Lambda_{11}^{-1} \\ \Lambda &= \Sigma^{-1} = \begin{pmatrix} \Lambda_{11} & \Lambda_{12} \\ \Lambda_{21} & \Lambda_{22} \end{pmatrix}. \end{aligned}$$

The *Bayes rule for Gaussians* is: if

$$\begin{aligned} z &\sim N(\mu_z, \Sigma_z) \\ y|z &\sim N(Wz + b, \Sigma_y) \end{aligned}$$

then $z|y \sim N(\mu_{z|y}, \Sigma_{z|y})$, where

$$\begin{aligned} \mu_{z|y} &= \Sigma_{z|y}[W'\Sigma_y^{-1}(y - b) + \Sigma_z^{-1}\mu_z] \\ \Sigma_{z|y}^{-1} &= \Sigma_z^{-1} + W'\Sigma_y^{-1}W \end{aligned}$$

(the Gaussian distribution is conjugate prior for the Gaussian likelihood).

The “completing the square” trick is not limited to scalars:

$$\begin{aligned} f(x) &= x'Ax + x'b + c \\ &= (x - h)'A(x - h) + k \\ h &= -\frac{1}{2}A^{-1}b \\ k &= c - \frac{1}{4}b'A^{-1}b \end{aligned}$$

The *exponential family density* is

$$\begin{aligned} p(y) &= h(y) \cdot \exp[\eta'T(y) - A(\eta)] \\ h: &\text{scale, e.g., } 1 \\ \eta: &\text{natural parameters, } t \\ T: &\text{sufficient statistic, e.g., id} \\ A: &\text{log-partition function (convex).} \end{aligned}$$

These are maximum entropy distributions, solutions of

$$\begin{aligned} \text{Find } & p \\ \text{To maximize } & H(p) \text{ (or } -\text{KL}(p||h)) \\ \text{Such that } & \mathbb{E}_{y \sim p}[T(y)] = t. \end{aligned}$$

The log-partition function A is the cumulant generating function

$$\begin{aligned} K(t) &= \log \mathbb{E}[e^{tX}] \\ \kappa_n &= K^{(n)}(0). \end{aligned}$$

In particular,

$$\begin{aligned} \nabla A(\eta) &= \mathbb{E}[T(y)] \\ \nabla^2 A(\eta) &= \text{Cov}[T(y)] \succcurlyeq 0 \end{aligned}$$

and A is convex.

4. The *maximum likelihood estimator* (MLE) is

$$\begin{aligned} \hat{\theta}_{\text{MLE}} &= \text{Argmax}_{\theta} \text{KL}(\text{data}||p_{\theta}) \\ &= \text{Argmax}_{\theta} H(\text{data}, p_{\theta}) \quad (\text{cross-entropy}) \\ &= \text{Argmin}_{\theta} \text{NNL}(\theta) \quad (\text{negative log-likelihood}). \end{aligned}$$

For binary classification, use a *surrogate loss function* (convex, decreasing, and above $\mathbf{1}_{[z \leq 0]}$) such as

$$\begin{aligned} \text{hinge} &: \text{ReLU}(1 - m) \\ \text{log} &: \log(1 + e^{-m}) \\ \text{exp} &: \exp(-m) \end{aligned}$$

where $m = \hat{y}y$, $y \in \{\pm 1\}$, $\hat{y} \in \mathbf{R}$.

The *method of moments* is an alternative to MLE; it may give invalid results.

Regularization with \log_{prior} is an equivalent to MAP (maximum a posteriori). Estimate the strength of the regularization with cross-validation.

The “one standard error rule” suggests to take the simplest model whose risk is no more than one standard deviation above that of the best model.

The *Bayesian* approach views probability in terms of *information*, rather than *repeated trials*.

Mixtures of conjugate distributions are still conjugate.

“Uninformative” priors are informative: call them “diffuse” instead.

To estimate the parameters and hyperparameters, one can use MLE, MAP, or a Bayesian approach – there is a hierarchy of increasingly Bayesian approaches:

Method	Parameters	Hyper-parameters
ML	ML	none
MAP	MAP	fixed
ML-II (empirical Bayes)	Bayes	ML
MAP-II	Bayes	MAP
Full Bayes	Bayes	Bayes

To summarize the posterior distribution, provide a (centered) *credible interval* or the *highest posterior density* (HPD) region (highest density interval, HDI).

If exact posterior inference is not possible, use approximations such as

– Laplace approximation

$$\begin{aligned} \int f(\theta) d\theta &\approx \frac{(2\pi)^{n/2}}{\sqrt{\det H}} f(\theta_{\text{Max}}) \\ \theta_{\text{Max}} &= \text{Argmax}_{\theta} f \\ H &= -\nabla^2 \log f|_{\theta=\theta_{\text{Max}}} \end{aligned}$$

obtained by approximating f with a Gaussian, *i.e.*, using a Taylor expansion of f at its mode; use a change of variable to make f more symmetric and closer to a Gaussian;

– Variational inference;
– MCMC (HMC).

In frequentist statistics (the data is a random variable, the parameters are fixed), the MLE is asymptotically Gaussian

$$\hat{\theta}_{\text{MLE}} \xrightarrow{d} N(\theta^*, (NF(\theta^*))^{-1})$$

where the *Fisher information matrix* is

$$\begin{aligned} F(\theta) &= \mathbb{E}_{X \in p_\theta} [(\nabla \log p_\theta(X))(\nabla \log p_\theta(X))^\top] \\ &= - \mathbb{E}_{X \sim p_\theta} [\nabla^2 \log p_\theta(X)] \\ &= \text{Hessian of the NNL.} \end{aligned}$$

There is a trade-off between bias and variance.

$$\begin{aligned} \text{bias} &= \mathbb{E}[\hat{\theta}] - \theta^* \\ V[\hat{\theta}] &= \mathbb{E}[\hat{\theta}^2] - \mathbb{E}[\hat{\theta}]^2 \\ \mathbb{E}[\cdot] &= \mathbb{E}_{\mathcal{D} \sim p_{\mathcal{D}}} [\cdot] \end{aligned}$$

5. The loss function we minimize should include the cost of incorrect output (and the reward of correct ones). For instance, classification problems could have a *reject option* (the algorithm is allowed to output “I do not know”).

$$\begin{aligned} (p_1, \dots, p_n) &\sim \text{Dirichlet}(1, \dots, 1) \\ j &\sim \text{Cat}([1, \dots, n], [p_1, \dots, p_n]) \\ i &= \underset{i}{\text{Argmax}} p_i \\ i &= p_i > \lambda ? i : 0 \\ \text{cost} &= \begin{cases} 0 & \text{if } i = j \\ \lambda_{\text{reject}} & \text{if } i = 0 \\ \lambda_{\text{error}} & \text{otherwise} \end{cases} \end{aligned}$$

The threshold λ minimizing $\mathbb{E}[\text{cost}]$ is

$$1 - \frac{\lambda_{\text{reject}}}{\lambda_{\text{error}}}.$$

In case of class imbalance, prefer the precision-recall curve to the ROC curve.

The *interpolated precision* for a given recall level α is the maximim precision of a recall level at least α (the precision is not necessarily decreassing with α). The *average precision* is the average of the interpolated precisions.

The F_β score is the weighted harmonic mean of precision and recall, with weights 1 and β^2 ; if recall (resp. precision) is more important, try $\beta = 2$ (resp. $\beta = 1/2$).

To choose between two models, in a Bayesian way, one can look at the *Bayesian factor* (for Bayesian hypothesis testing):

$$\begin{aligned} \frac{P(M_1|D)}{P(M_2|D)} &= \frac{P(M_1, D)/P(D)}{P(M_2, D)/P(D)} \\ &= \frac{P(D|M_1)P(M_1)}{P(D|M_2)P(M_2)} \\ B &= \frac{P(D|M_1)}{P(D|M_2)} \text{ if } P(M_1) = P(M_2) = \frac{1}{2}. \end{aligned}$$

These are *marginal likkelihoods* (or “evidence”): the parameters have been integrated out,

$$p(D|M) = \int p(D|M, \theta)p(\theta|M)d\theta.$$

The marginal likelihood can be difficult to compute: by approximating the posterior with a Gaussian (Laplace approximation)

$$\begin{aligned} \log p(D|M) &\approx \log p(D|\hat{\theta}) + \log p(\hat{\theta}) - \frac{1}{2} \log |H| \\ H &= -\nabla^2 \log p(D, \theta)|_{\theta=\hat{\theta}}, \end{aligned}$$

choosing a uniform prior $p(\theta) \propto 1$, and approximating the Occam factor

$$\begin{aligned} H &= \sum_i \nabla^2 \log p(D_i|\theta) \\ &= \sum_i H_i \\ &\approx N \cdot \hat{H} \\ \log |H| &\approx \log |N \hat{H}| \\ &= D \log N + \log |\hat{H}| \\ &\approx D \log N \end{aligned}$$

(because $\log |\hat{H}|$ is constant and becomes negligible when $D \log N \rightarrow \infty$), we get the Bayesian information criterion (BIC)

$$\log p(d|m) \approx \log p(D, \hat{\theta}, m) - \frac{D_m}{2} \log N$$

(it is sometimes multiplied by -2 to get a loss). Alternatives to the BIC include:

$$\begin{aligned} \text{AIC} &= -2 \log p(D|\hat{\theta}, m) + 2D \\ \text{MDL} &= -\log p(D|\hat{\theta}, m) = \log p(m) \end{aligned}$$

where $p(m)$ is the number of bits needed to specify which model m it is.

Point null hypotheses, $H_0 : \mu = 0$, are irrelevant: eeplace 0 with a *region of practical equivalence* (ROPE) $H_0 : \mu \in [-\varepsilon, \varepsilon]$; we can then compute $P(H_0|\text{Data})$ (Bayesian T test).

The (frequentist) risk of an estimator $\hat{\theta}$ is

$$\text{Risk}(\theta, \hat{\theta}) = \mathbb{E}_{X \sim p_\theta} [\ell(\theta, \hat{\theta}(X))].$$

The *Bayesian risk* integrates out the (unknown) true parameter θ with a prior.

In supervised learning, the population risk is

$$\text{Risk}(f, p^*) = \mathbb{E}_{x, y \sim p^*} [\ell(y, f(x))]$$

where (x, y) are sampled from the true (unknown) joint distribution p^* . The empirical risk replaces p^* with the empirical distribution of the observed data

$$\text{Risk}(f, \text{Data}) = \mathbb{E}_{x, y \sim \text{Data}} [\ell(y, f(x))].$$

The empirical risk can be decomposed into

- The approximation error (risk of the best model in the class considered);
- The estimation error (generalization gap), stemming from the difference between the population distribution and the empirical distribution.

Add a regularization to the risk

$$\text{Risk}(f, \text{Data}) + \lambda C(f)$$

and choose λ to minimize the LOO-CV risk.

Frequentist hypothesis testing is based on a fallacy:

If H_0 were true, this statistic would probably not occur.

This statistic did occur.

Therefore, H_0 is probably false.

If a person is American, he is probably not a member of congress.

This person is a member of congress.

Therefore, he is probably not American.

6. Joint entropy, mutual information, conditional entropy behave like set union, intersection and difference.

$$H(p) = \mathbb{E}_{X \sim p} -\log_2 p(x) \quad \text{entropy}$$

$$H_{ce}(p, q) = \mathbb{E}_{X \sim p} -\log_2 q(x) \quad \text{cross-entropy}$$

$$H(X, Y) = -\sum_{x, y} p(x, y) \log_2 p(x, y) \quad \text{joint entropy}$$

$$H(Y|X) = \mathbb{E}_{x \sim p_X} H(Y|X = x) \quad \text{conditional entropy}$$

$$= H(X, Y) - H(X)$$

$$H(X_1, \dots, X_n) = \sum H(X_i | X_1, \dots, X_{i-1}) \quad \text{chain rule}$$

$$\text{perplexity}(p) = 2^{H(p)} \quad \text{average branching factor}$$

$$\text{KL}(p||q) = \mathbb{E}_{X \sim p} \left[\log \frac{p(x)}{q(x)} \right] \quad \text{relative entropy}$$

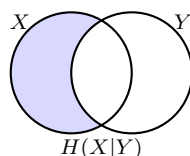
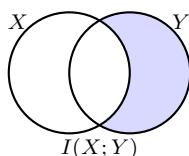
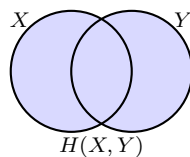
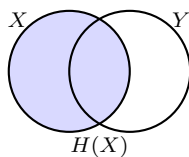
$$= -H(p) + H_{ce}(p, q)$$

$$I(X; Y) = \text{KL}(p(x, y) || p(x)p(y)) \quad \text{mutual information}$$

$$= H(X) - H(X|Y)$$

$$= H(X, Y) - H(X|Y) - H(Y|X)$$

$$= H(X) + H(Y) - H(X, Y)$$



$$I(X; Y|Z) = \mathbb{E}_{z \sim p_Z} [I(X; Y)|Z = z]$$

$$= I(Y; X, Z) - I(Y; Z)$$

$$I(Z_1, \dots, Z_n | X) = \sum I(Z_i; X | Z_1, \dots, Z_{i-1})$$

$$\text{NMI}(X, Y) = \frac{I(X; Y)}{\text{Min}(H(X), H(Y))} \leq 1$$

$$X \rightarrow Y \rightarrow Z \text{ Markov} \implies X \perp\!\!\!\perp Z | Y$$

$$\implies I(X; Y) \geq I(X; Z)$$

7. The *induced norm* of a matrix $A \in \mathbf{R}^{m \times n}$ is

$$\|A\|_p = \text{Max}_{\|x\|=1} \|Ax\|_p;$$

in particular,

$$\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)} = \text{Max}_i \sigma_i$$

is the largest singular value.

The *nuclear norm* (or trace norm) is

$$\|A\|_* = \text{tr} \sqrt{A^T A} = \sum_i \sigma_i = \sum_i |\sigma_i| = \|\sigma\|_1;$$

as an L^1 norm, it has a sparsifying effect on the singular values: it encourages low-rank matrices. The *Shatten p norm* generalizes it:

$$\|A\|_p = (\sum_i \sigma_i^p)^{1/p}.$$

The *Frobenius norm* is the L^2 norm of the entries of a matrix,

$$\|A\|_F^2 = \|\text{vec } A\|_2^2 = \text{tr}(A^T A) = \mathbb{E}_{v \sim N(0, I)} \|Av\|_2^2.$$

To invert (or compute the determinant) of a partitioned matrix $\begin{pmatrix} * & * \\ * & * \end{pmatrix}$, pre-multiply it by $\begin{pmatrix} I & * \\ 0 & I \end{pmatrix}$ to get $\begin{pmatrix} * & 0 \\ * & * \end{pmatrix}$, and post-multiply the result with $\begin{pmatrix} I & 0 \\ * & I \end{pmatrix}$ to get to get $\begin{pmatrix} * & 0 \\ 0 & * \end{pmatrix}$, which is easier to invert; the result contains *Schur complements* $E - FH^{-1}G$.

$$\begin{bmatrix} I & -FH^{-1} \\ 0 & I \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} \begin{bmatrix} I & 0 \\ -H^{-1}G & I \end{bmatrix} = \begin{bmatrix} E - FH^{-1}G & 0 \\ 0 & H \end{bmatrix}$$

Using instead $\begin{pmatrix} I & 0 \\ * & I \end{pmatrix}$ and $\begin{pmatrix} I & * \\ 0 & I \end{pmatrix}$ and equating the resulting formulas gives the *Woodbury formula* (inverse of a Schur complement)

$$(E - FH^{-1}G)^{-1} = E^{-1} + E^{-1}F(H - GE^{-1}F)^{-1}GE^{-1}$$

(the matrix to invert on the rhs can have a different, smaller dimension).

Applications include

$$(\Sigma + \underset{N \times N}{XX'})^{-1} = \Sigma^{-1} - \Sigma^{-1}X(\underset{D \times D}{I + X'\Sigma^{-1}X})^{-1}X'\Sigma^{-1},$$

the rank-1 update

$$(A + uv')^{-1} = A^{-1} - \frac{A^{-1}uv'A^{-1}}{1 + v'A'u}$$

and the distribution of a conditional Gaussian

$$\begin{aligned} X_1|X_2 &\sim N(\bar{\mu}, \bar{\Sigma}) \\ \bar{\mu} &= \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(x_2 - \mu_2) \\ \bar{\Sigma} &= \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}. \end{aligned}$$

PCA whitening

$$\begin{aligned} \Sigma &= \frac{1}{N}X'X = EDE' \quad (\text{eigendecomposition}) \\ X &= USV' \quad (\text{SVD}) \\ W_{\text{pca}} &= D^{-1/2}E' \\ Y &= \frac{X}{N \times D} \frac{W_{\text{pca}}}{D \times D} \end{aligned}$$

ensures $\text{Cov } Y = I$, but we can add any rotation matrix; *Mahalanobis whitening* keeps the transformed data as close as possible to the original

$$W_{\text{zca}} = ED^{-1/2}E' = \Sigma^{-1/2} = VS^{-1}V'.$$

The Schur complement of E in $\begin{bmatrix} E & F \\ G & H \end{bmatrix}$ is

$$\begin{aligned} \overline{E} &= \overline{H} - \overline{G} \overline{E}^{-1} \overline{F} \\ \bar{E} &= H - GE^{-1}F. \end{aligned}$$

Similarly,

$$\begin{aligned} \overline{H} &= \overline{E} - \overline{F} \overline{H}^{-1} \overline{G} \\ \bar{H} &= E - FH^{-1}G. \end{aligned}$$

8. Gradient descent, with learning rate $\eta < 2/L$, where L is the Lipschitz constant of the gradient, is guaranteed to converge,

$$\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(\theta_t).$$

The *Armijo-Goldstein condition*, for the stepsize ensure sufficient reduction of the objective function without performing an exact line search,

$$\ell(\theta_t + \eta d_t) \leq \mathcal{L}(\theta_t) + c\eta d_t^\top \nabla \mathcal{L}(\theta_t),$$

where d_t is the descent direction, e.g., $-\nabla \mathcal{L}(\theta_t)$, and $c = 10^{-4}$.

Momentum methods (heavy ball, Nesterov) modify the descent direction using past (or extrapolated) gradients.

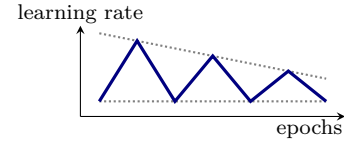
Second order methods approximate Newton's method

$$\theta_{t+1} = \theta_t - \eta (\nabla^2 \mathcal{L}(\theta_t))^{-1} \nabla \mathcal{L}(\theta_t)$$

by using approximations of the Hessian (BFGS) or adding constraints (trust region, if the Hessian is not positive semi-definite, if the objective function is not convex).

To choose the learning rate, check the loss after a few epochs, for several learning rates (e.g., in $[10^{-5}, 10^1]$). Common learning rate schedules include

- Exponential decay;
- Polynomial decay;
- One cycle (linear warm-up, followed by cosine cool-down);
- Cyclical learning rate.



Stochastic variance reduced gradient (*SVRG*) uses a control variate: a baseline value of the gradient, computed on the whole dataset, updated once in a while.

$$g_t = \nabla \mathcal{L}(\theta_t) - \nabla \mathcal{L}_t(\tilde{\theta}) + \nabla \mathcal{L}(\tilde{\theta})$$

SAGA is similar, but keeps track of the latest gradient of each minibatch to progressively update the gradient.

Preconditioned SGD

$$\theta_{t+1} = \theta_t - \eta M_t^{-1} g_t$$

includes AdaGrad, RMSProp, AdaDelta, Adam; non-diagonal preconditioners include full-matrix AdaGrad

$$\begin{aligned} M_t &= [(G_t G_t^\top)^{1/2} + \varepsilon T]^{-1} \\ G_t &= [g_1, \dots, g_1] \end{aligned}$$

and *Shampoo* (a block-diagonal, Kronecker approximation).

The constrained optimization problem

$$\text{Maximize}_{\theta} \mathcal{L}(\theta) \text{ such that } g(\theta) \leq 0, h(\theta) = 0$$

can be written

$$\text{Minimize}_{\theta} \text{Max}_{\mu \geq 0, \lambda} \underbrace{\mathcal{L}(\theta) + \mu^\top g(\theta) + \lambda^\top h(\theta)}_{L(\theta, \mu, \nu): \text{generalized Lagrangian}}$$

If \mathcal{L} and g are convex, the critical points satisfy the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{aligned} g(\theta) &\leq 0, \quad h(\theta) = 0 && \text{feasibility} \\ \nabla \mathcal{L}(\theta) + \sum \mu_i \nabla g_i(\theta) + \sum \lambda_j \nabla h_j(\theta) &= 0 \\ \mu &\geq 0 && \text{dual feasibility} \\ \mu \odot g &= 0 \end{aligned}$$

To optimize objective functions of the form

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{smooth}}(\theta) + \mathcal{L}_{\text{rough}}(\theta),$$

the *proximal gradient* method uses

$$\theta_{t+1} = \text{prox}_{\eta \mathcal{L}_{\text{rough}}} [\theta_t - \eta \nabla \mathcal{L}_{\text{smooth}}(\theta_t)]$$

where the proximal operator is

$$\begin{aligned} \text{prox}_{\eta \mathcal{L}}(\theta) &= \underset{z}{\text{Argmin}} \left(\mathcal{L}(z) + \frac{1}{2\eta} \|z - \theta\|_2^2 \right) \\ &= \underset{z}{\text{Argmin}} \mathcal{L}(z) \text{ such that } \|z - \theta\|_2 \leq \rho \end{aligned}$$

(where ρ depends on η). Examples include

- Projected gradient (hard constraint $\theta \in C$, *i.e.*, $\mathcal{L}_{\text{rough}} = I_C$);
- L^1 regularization (soft thresholding)

$$\begin{aligned} \text{prox}_{\lambda \|\cdot\|_1}(\theta) &= \underset{x}{\text{Argmin}} \|x\| + \frac{1}{2\lambda} (x - \theta)^2 \\ &= \begin{cases} \theta - \lambda & \text{if } \theta \geq \lambda \\ 0 & \text{if } |\theta| \leq \lambda \\ \theta + \lambda & \text{if } \theta \leq -\lambda \end{cases} \end{aligned}$$

- Quantization

$$\mathcal{L}_{\text{rough}}(\theta) = \inf_{\theta_0 \in C} \|\theta - \theta_0\|_1,$$

where $C = \{\pm 1\}^D$ (the proximal operator is a generalization of soft-thresholding: ProxQuant).

The *MM algorithm* (minorize-maximize) maximizes a function $\ell(\cdot)$ by using a surrogate function $G(\cdot, \cdot)$ such that

$$\begin{aligned} Q(\theta, \theta') &\leq \ell(\theta) \\ Q(\theta, \theta) &= \ell(\theta) \end{aligned}$$

by iterating $\theta_{t+1} = \underset{\theta}{\text{Argmax}} Q(\theta, \theta_t)$.

The *EM algorithm* maximizes the likelihood of the observed data y by marginalizing the unobserved (latent, missing) data z

$$\ell(\theta) = \sum_n \log p(y_n | \theta) = \sum_n \log \sum_z p(y_n, z | \theta)$$

by using Jensen's inequality

$$\begin{aligned} \ell(\theta) &= \sum_n \log \sum_z q_n(z) \frac{p(y_n, z | \theta)}{q_n(z)} \\ &\geq \sum_n \sum_z q_n(z) \log \frac{p(y_n, z | \theta)}{q_n(z)} \\ &= \text{ELBO}(\theta, q_{1:N}) \end{aligned}$$

$$Q(\theta, \theta') = \text{ELBO}(\theta, p(z_n | y_n, \theta)_{n \in [1, N]})$$

It can be used to compute the MLE or MAP estimator of a Gaussian mixture model.

9. A generative classifier is a model of the form

$$p(y|x) \propto p(x|y)p(y)$$

(as opposed to a discriminative classifier, which models $p(y|x)$ directly). For instance, Gaussian discriminant analysis uses

$$X | Y = y \sim N(\mu_y, \Sigma_y);$$

if the covariance matrices Σ_y do not depend on y , this is *linear discriminant analysis* (LDA); if not, this is quadratic discriminant analysis (QDA).

Fisher linear discriminant analysis (FLDA) first reduces the dimension, choosing the projection giving the best classification possible.

The naive Bayes classifier assumes that the features are independent when conditioned on the class label

$$p(\mathbf{x}|y) = \prod_d p(x_d|y).$$

10. (Binary) logistic regression is a discriminative classification model

$$Y | X = x \sim \text{Bernoulli}(\sigma(w'x + b)).$$

The gradient and the Hessian of the log-likelihood can be computed explicitly; Newton's method can be formulated as an iteratively-reweighted least squares (IRLS) problem; *Fisher scoring* replaces the Hessian with its expectation, the Fisher information matrix.

(Multinomial) logistic regression can be fitted with *bound optimization* (aka the MM algorithm)

$$\begin{aligned} \ell(\theta) &= Q(\theta, \theta') \\ Q(\theta, \theta') &= \ell(\theta') + (\theta - \theta')^\top \nabla \ell(\theta') + \frac{1}{2} (\theta - \theta')^\top B (\theta - \theta') \\ \theta_{t+1} &= \theta_t - B^{-1} \nabla \ell(\theta_t) \end{aligned}$$

where $\forall \theta \ H(\theta) \succcurlyeq B$. For binary logistic regression, $B = -\frac{1}{4} X^\top X$ works.

Multinomial logistic regression $p(y|x) \propto \exp(w_y^\top x)$ can be generalized to *maximum entropy classifiers*

$$p(y|x) \propto \exp[w^\top \phi_y(x)]$$

(they are often used in NLP).

For consistent hierarchical classification (taxonomy), add mutual exclusion constraints between sibling label nodes.

If there is a large number of classes, try a hierarchical softmax, *i.e.*, a tree of binary classifiers structured, e.g., with Huffman coding.

To deal with class imbalance, try:

- Logit adjustment;
- Resampling, to make the data more balanced;

$$p_y \propto N_y^q \quad q \in [0, 1]$$

- Nearest mean classifier

$$f(x) = \underset{y}{\text{Argmin}} \left\| \phi(x) - \underset{i: y_i = y}{\text{Mean}} \phi(x_i) \right\|_2^2$$

where ϕ are learned features, from a DNN, trained on the original, unbalanced data.

To make logistic regression robust, *bitempered logistic regression*

- Uses *tempered cross-entropy* as loss function, to reduce the influence of mislabeled points far away from the decision boundary;
- Uses a *tempered softmax* to reduce the influence of mislabeled points close to the decision boundary.

For Bayesian logistic regression, use a Laplace approximation

11. Least squares regression can be fitted with:

- $\hat{w} = (X'X)^{-1}X'y$;
- SVD (more numerically stable);
- QR (faster if X is tall);
- Conjugate gradient (if $X \succcurlyeq 0$);
- GMRES (if X is sparse);

The gradient and Hessian of the residual sum of squares are

$$\begin{aligned}\nabla_w \text{RSS}(w) &= X'Xw - X'y \\ \nabla_w^2 \text{RSS}(w) &= X'X.\end{aligned}$$

Modeling (X, Y) as a joint Gaussian and conditioning on X is equivalent to linear regression (only for Gaussian models).

Ridge regression linear regression MAP) can be fitted with standard OLS regression by adding virtual data to account for the prior; use empirical Bayes

$$\lambda = \underset{\lambda}{\text{Argmax}} \log p(D|\lambda)$$

to choose the regularization parameter λ .

The *lasso* is a MAP estimator with a Laplace prior; it can lead to unstable results if some predictors are highly correlated (sometimes including one, sometimes the other) – the *elasticnet* addresses that problem (if there is a group of highly correlated predictors, it tends to include either none or all of them, with comparable weights: *soft grouping*).

The lasso can be solved with:

- Coordinate descent (soft-thresholding, one coordinate at a time);
- Projected gradient descent (after writing the lasso loss as a quadratic program, by splitting $w = w_+ - w_-$;
- Proximal gradient descent (ISTA);
- LARS (a continuation method, aka homotopy method) to get the whole regularization path: start with a large value of λ , so that the model only uses one predictor, progressively decrease λ (it is possible to find, analytically, the value of λ when the set of predictors used changes);
- LAR (a greedy simplification of LARS: it only adds predictors, but never removes them).

For *spline* regression, check **patsy.bs**. B-splines use a fixed number of knots. Smoothing splines use N knots if there are N data points and add an ℓ^2 regularization.

Generalized additive models (GAM) can be estimated with backfitting.

A Student (SGD, EM) or Laplace (LP) likelihood leads to robust regression. The Huber loss is faster.

RANSAC makes regression robust by fitting a model to a small subset of points, using it to remove outliers, and refitting the model on the inliers (with multiple restarts).

Bayesian linear regression generalizes ridge regression (which is just a MAP estimator).

MAP with an empirical Bayes prior on the regression coefficients results in a sparse estimate \hat{w} (automatic relevancy determination, ARD, sparse Bayesian learning).

12. A GLM is a conditional exponential family

$$\begin{aligned}p(y|x, w, \sigma) &\propto h(y, \sigma^2) \exp \frac{\langle y, w'x \rangle}{\sigma^2} \\ &= h(y, \sigma^2) \exp \frac{\langle y, \eta \rangle - A(\eta)}{\sigma^2} \\ \eta &= w'x \text{ (natural parameter)}\end{aligned}$$

$$E[y|x, w, \sigma^2] = A'(\eta) = \ell^{-1}(\eta)$$

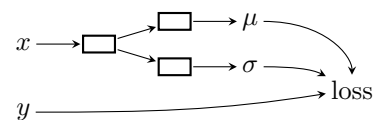
$$\text{Var}[y|x, w, \sigma^2] = A''(\eta)\sigma^2$$

ℓ : link function

13. The perceptron used a (non-differentiable) Heaviside activation function.

MLPs can be used for

- Heteroskedastic regression;



- Classification (tabular/image/text data);
- etc.

Activation functions include: ReLU, leaky ReLU, ELU, SELU, Swish (aka SiLU), GELU, etc.

Initialization schemes include Xavier (logistic), He (ReLU), LeCun, LSUV (layer sequential unit variance: start with orthogonal weight matrices, and rescale them so that the activations have unit variance on the first minibatch).

Regularization includes early stopping, weight decay, sparsity (but prefer block-sparse matrices: GPUs are optimized for dense matrix operations), dropout, Bayesian neural nets, (stochastic) gradient descent (implicit regularization).

Other feed-forward networks include:

- RBF networks:

$$\begin{aligned}y|x &\sim N(w'\phi_\mu(x), \sigma_y^2) \\ \phi_\mu(x) &= [k(x, \mu_1), \dots, k(x, \mu_m)] \\ k(x, \mu) &= \exp -\frac{1}{2\sigma_x^2} \|x - \mu\|^2\end{aligned}$$

(if the μ_i 's are fixed, this is a linear regression);

- Mixtures of experts (for multi-modal output distributions, one-to-many functions)

$$z \sim \text{Cat}(\text{Softmax } f(x))$$

$$y \sim N(\mu_z(x), \sigma_z^2(x))$$

16. Distance-based methods, such as *k-nearest neighbours*, suffer from the curse of dimensionality: in high dimension, the nearest point is far away, and all pairwise distances tend to be the same. Efficient implementations can use k-d trees or LSH (in Python, check `faiss`).

Metric learning searches for a Mahalanobis matrix M such that the distance

$$d(x, x') = \sqrt{(x - x')^\top M (x - x')}$$

works well when used for *k*-nn classification (or such that samples in the same class are close and samples in different classes are far apart).

This can be solved with semidefinite programming. Alternatively, parametrize M as $M = w'w$.

The probability that j is the nearest neighbour of i is

$$p_{ij} \propto \exp - \|Wx_i - Wx_j\|_2^2.$$

We can choose W such that it maximizes the expected number of samples correctly classified with 1-nn (neighbourhood component analysis, NCA). Latent coincidence analysis (LCA) estimates the model

$$Z \sim N(Wx, \sigma^2 I)$$

$$Z' \sim N(Wx', \sigma^2 I)$$

$$P[y = y'] = \exp - \|z - z'\|^2$$

Deep metric learning (DML) also learns an embedding and computes the distances in embedding space (you can then remove M).

Losses for DML include

- Pairwise contrastive loss (hinge loss);
- Triplet loss (may require hard negative mining);
- *n*-pairs loss (InfoNCE).

The word “kernel” has several (sometimes overlapping) meanings in statistics:

- A *density kernel* is a function $\mathbf{R}^n \times \mathbf{R}^n \rightarrow \mathbf{R}$ with non-negative values, used for smoothing (kernel density estimation (KDE), kernel regression)
- A *positive definite kernel* is a function $\kappa : \mathbf{R}^n \times \mathbf{R}^n \rightarrow \mathbf{R}$, such that

$$\forall l \forall x_1, \dots, x_m \in \mathbf{R}^n \quad (\kappa(x_i, x_j))_{i \leq l, j \leq m} \succcurlyeq 0$$

is positive definite; it is used for the “kernel trick”;

- The transition kernel of a Markov chain;
- A convolution kernel;
- etc.

17. Positive definite kernels (aka *Mercer kernels*) correspond to feature maps: for structured objects

(graphs, text, etc.), it may be easier to define kernels than features (string kernel, random walk kernel).

Random features can approximate the RBF kernel.

18. Classification and regression trees (CART) are fitted greedily, splitting the nodes by looking at:

- The Gini index (a generalized entropy – the Herfindahl index);
- The entropy (aka deviance).

Bagging uses random subsets of the data. *Random forests* also use random subsets of the variables. *Boosting* successively fits models on data weighted by the errors made by the best linear combination of the models so far. Bagging and random forests reduce the variance; boosting reduces the bias.

19. Empirical risk minimization (ERM) minimizes the expected loss wrt the empirical distribution – but a sum of Dirac masses is unlikely to be a good approximation of the population distribution. Data augmentation attempts to address this problem (vicinal risk minimization).

Transfer learning adapts a pretrained model

- By fine-tuning it;
- By replacing its last layers;
- By adding adjustment layers (“adapters”) inside.

Self-supervised learning (SSL) often uses

- Imputation tasks (fill-in-the-blank);
- Proxy tasks, in particular *contrastive tasks*: attempting to tell if two samples are related (e.g., augmentations of the same sample, or in the same class) or not (SimCLR).

In *transfer learning*, the input is similar, but the output is different. In *domain adaptation*, the inputs are different, but the output labels are the same. For instance, domain adversarial learning trains a classifier whose latent representation is unable to distinguish between the source and target domains.

Self-training with *pseudo-labels* (using predictions as labels for unlabeled data) suffers from *confirmation bias*: only keep pseudo-labels the model is confident in. Co-training and tri-training estimate that confidence by training several models, either on (independent) sets of features, or on subsets of the data.

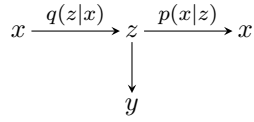
Entropy minimization is another for of semi-supervised learning, which implements the *cluster assumption*: the decision boundary should be in a low-density region of the data manifold.

Label propagation is another semi-supervised learning technique: build a similarity graph of samples, and propagate the labels from the labeled samples.

Data augmentation can also be applied to unlabeled data; perturbations of a data point should not cause a large change in the output (*consistency regularization*).

Deep generative models, such as VAEs, can also be

leveraged for semi-supervised learning.



You can also train the discriminant of a GAN to output “fake” or the class label (instead of “fake”/“real”).

20. To select the number of principal components to retain, some suggest to look at the maximum of the *profile likelihood*: the likelihood of a Gaussian $N(\mu_1, \sigma^2)$ on $\lambda_1 \geq \dots \geq \lambda_L$ and $N(\mu_2, \sigma^2)$ on $\lambda_{L+1} \geq \dots \geq \lambda_n$ (the two Gaussian have the same variance).

Factor analysis (FA)

$$\begin{aligned} z &\sim N(\mu_0, \Sigma_0) \\ x &= Wz + \mu + \varepsilon \\ \varepsilon &\sim N(0, \Psi) \end{aligned}$$

can be computed with the EM algorithm.

Probabilistic PCA (PPCA) is the special case $\Psi = \sigma^2 I$. FA can be generalized: non-linear FA, mixture FA, exponential family FA, etc.

For paired data, check

– Supervised PCA

$$\begin{aligned} z &\sim N(0, I) \\ x &= W_1 z + \varepsilon & \varepsilon &\sim N(0, \sigma_1^2 I) \\ y &= W_2 z + \varepsilon & \eta &\sim N(0, \sigma_2^2 I) \end{aligned}$$

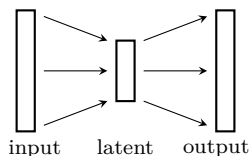
– Partial least squares

$$\begin{aligned} z_0 &\sim N(0, I) \\ z_1 &\sim N(0, I) \\ x &= W_1 z_0 + B z_1 + \varepsilon & \varepsilon &\sim N(0, \sigma_1^2 I) \\ y &= W_2 z_0 + \varepsilon & \eta &\sim N(0, \sigma_2^2 I) \end{aligned}$$

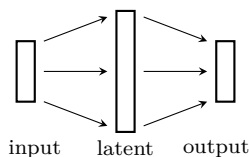
– Canonical correlation analysis

$$\begin{aligned} z_0 &\sim N(0, I) \\ z_1 &\sim N(0, I) \\ z_2 &\sim N(0, I) \\ x &= W_1 z_0 + B_1 z_1 + \varepsilon & \varepsilon &\sim N(0, \sigma_1^2 I) \\ y &= W_2 z_0 + B_2 z_2 + \varepsilon & \eta &\sim N(0, \sigma_2^2 I) \end{aligned}$$

Auto-encoders are non-linear generalizations of PCA; the latent representation is usually a narrow bottleneck layer



but one could use a larger layer, if it is regularized (noise, sparsity, etc.)



There are many *manifold learning* algorithms:

– Multidimensional scaling (MDS – actually PCA)

$$\text{Minimize}_z \sum (\langle \tilde{x}_i, \tilde{x}_j \rangle - \langle \tilde{z}_i, \tilde{z}_j \rangle)^2$$

where the \tilde{x}_i 's are the centered data points;

– Metric MDS

$$\text{Minimize}_z \sum (\|x_i - x_j\| - \|z_i - z_j\|)^2$$

– Non-metric MDS

$$\text{Minimize}_{f \text{ monotonic}, z} \frac{\sum [f(\|x_i - x_j\|) - \|z_i - z_j\|]^2}{\sum \|z_i - z_j\|^2}$$

– Sammon (metric MDS with more weight on shorter distances)

$$\text{Minimize}_z \sum \frac{(\|x_i - x_j\| - \|z_i - z_j\|)^2}{\|x_i - x_j\|}$$

– Isomap is MDS on the k -nearest neighbour graph of the datapoints

– Kernel PCA (kPCA) is PCA on the Gram matrix for some kernel

– Maximum variance unfolding (MVU)

$$\begin{aligned} \text{Find } z \\ \text{To maximize } & \sum \|z_i - z_j\|^2 \\ \text{Such that } & \forall (i, j) \in G \quad \|z_i - z_j\| = \|x_i - x_j\| \end{aligned}$$

where G is the k -NN graph (this can be formulated as a semidefinite program)

– Local linear embedding (LLE)

$$\text{Minimize}_z \sum_i \left\| z_i - \sum_j w_{ij} z_j \right\|^2$$

where the w_{ij} are the barycentric coordinates of x_i in the k -nn graph (W is sparse)

$$W = \underset{w}{\text{Argmin}} \sum \left\| x_i - \sum_{i \sim j} w_{ij} x_j \right\|^2 \text{ st } \forall i \sum w_{ij} = 1$$

– Laplacian eigenmaps

$$\text{Minimize}_z \sum W_{ij} \|z_i - z_j\|^2 \text{ st } W^\top D Z = I$$

where

$$W_{ij} = \exp - \frac{\|x_i - x_j\|^2}{2\sigma^2}$$

if $i \sim j$ is in the k -nn graph (this is equivalent to the generalized eigenvalue problem $Lz_i = \lambda_i D z_i$, where $L = D - W$, and D is the degree matrix $D_{ii} = \sum_{ij} W_{ij}$).

Stochastic neighbour embedding (SNE) replaces distances with probability distributions

$$p_{j|i} \propto \exp - \frac{\|x_i - x_j\|^2}{2\sigma_i^2}$$

(the probability that j is a neighbour of i) and looks for points z_i with similar probability distributions

$$q_{j|i} \propto \exp - \|z_i - z_j\|^2$$

(the variance is fixed) as measured by the KL divergence

$$\text{Minimize}_z \sum_i \text{KL}(p_i \| p_j) = \sum_{ij} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

Symmetric SNE uses

$$\text{Minimize}_z \text{KL}(p \| q)$$

where

$$p_{ij} \propto \exp - \frac{\|x_i - x_j\|^2}{\sigma^2}.$$

T-distributed SNE (t-SNE) uses a Cauchy distribution (Student T with 1 degree of freedom) instead of a Gaussian to limit over-crowding

$$q_{ij} \propto \frac{1}{1 + \|z_i - z_j\|^2}.$$

The scales σ_i are chosen so that the p_i 's have a user-specified perplexity (*i.e.*, entropy), which can be interpreted as the effective number of neighbours.

UMAP is faster than t-SNE and tends to preserve the global structure better.

Latent semantic indexing (LSI) is a low-rank factorization of the term-document (or tf-idf) matrix.

Latent semantic analysis (LSA) is the SVD of the word co-occurrence matrix, or the PMI (pointwise mutual information) matrix

$$\text{PMI}_{ij} = \log \frac{p(i, j)}{p(i)p(j)} \quad \text{or} \quad \text{Max}(\text{PMI}_{ij}, 0).$$

GloVe is a simplified, faster alternative to word2vec's skipgram.

21. There is no satisfactory way of assessing the quality of a clustering:

- Purity = $\sum_i \frac{N_i}{N} p_i$, where N_i is the size of cluster i and p_i its purity, *i.e.*, the proportion of items belonging to the majority class (but this does not penalize the number of clusters);
- Rand index: accuracy of the classifier of pairs, using “are in the same cluster” as a predictor of “are in the same class”;
- Adjusted Rand index, which compares the Rand index with its expected value;
- Mutual information (but this does not penalize the number of clusters);

– Normalized mutual information

$$\text{NMI}(U, V) = \frac{I(U; V)}{\frac{1}{2}[H(U) + H(V)]}.$$

To estimate the number of clusters, one could look at the distortion, the silhouette, or (better) the BIC (from the log marginal likelihood) of a probabilistic model, *e.g.*, a GMM (Gaussian mixture model).

The eigenspace, for the eigenvalue 0, of the (weighted) Laplacian of a graph is spanned by the indicator vectors of its connected component. *Spectral clustering* is k -means in the span of the eigenvectors of the Laplacian of the similarity matrix with the smallest eigenvalues (prefer the normalized Laplacian, $L_{\text{sym}} = D^{-1/2} L D^{-1/2}$). It is a relaxation of the normalized cut problem.

22. Recommender systems can use:

- Matrix factorization (fitted with alternating least squares, ALS, or SGD, which handles missing values);
- Auto-encoders $\hat{Y} = W^\top \phi(VY)$;
- Factorization machines

$$x = [\text{one-hot}(u), \text{one-hot}(i)]$$

$$f(x) = \mu + \sum w_i x_i + \sum_{i < j} (v_i^\top v_j) x_i x_j$$

– Neural matrix factorization

$P_{u\cdot}, U_{u\cdot}$: user embeddings

$Q_{i\cdot}, V_{i\cdot}$: item embeddings

$$z_{ui}^1 = P_{u\cdot} \odot Q_{i\cdot}$$

$$z_{ui}^2 = \text{MLP}([U_{u\cdot}, V_{i\cdot}])$$

$$\hat{Y}_{ui} = \sigma(w^\top [z_{ui}^1, z_{ui}^2])$$

They can also leverage implicit feedback (if a user did not rate an item, they were not really interested in it).

23. To compute node embeddings, try to reconstruct the graph, using:

- A dot-product decoder, $\hat{w} = zz^\top$;
- A distance decoder, $\hat{w}_{ij} = d(z_i, z_j)$, possibly with a Poincaré (hyperbolic) distance,

$$d(z_i, z_j) = \text{arccosh} \left[1 + \frac{2 \|z_i - z_j\|^2}{(1 - \|z_i\|^2)(1 - \|z_j\|^2)} \right].$$

For directed graphs, learn two embeddings per node: a source and a target embedding.

DeepWalk, Node2Vec use a skipgram method on random walks to compute node embeddings.

**Getting more for less:
better A/B testing via causal regularization
K. Webster (2022)**

Price changes can be decomposed into market impact (caused by your trade) and alpha (everything else).

You cannot estimate trade impact from actual trades: they have alpha. You can estimate them from random (alpha-less) trades but, since they are costly, you have very few of them. Instead, estimate trade impact on actual trades, but with a regularized model (e.g., ridge regression), and use random trades to select the regularization parameter

***Investment decisions
under almost complete causal ignorance***
J. Simonian (2022)

The *probability distance* between two DAGs is the number of conditional independencies entailed by one and not by the other (normalized by the number of possible independencies). The *counterfactual distance* is the number of causes (pairs of nodes (X, Y) , such that there exists a causal path $X \rightarrow \dots \rightarrow Y$) present in one but not in the other (normalized by the number of possible causes). Take the sum of those two distances.

If you hesitate between several causal graphs, you could pick that minimizing $\lambda_i = \sum_{j \neq i} d(G_i, G_j)$. Alternatively, you could build a strategy for each of them, investing w_{ik} in asset k , and build a portfolio, with weights $\sum_i w_{ik} x_k$, maximizing $\sum_{ik} \lambda_i w_{ik} x_k$ (and subject to constraints of your choice).

***Addendum on how many times
cointelated pairs cross paths***
B. Mahdavi-Damghani and S. Roberts (2020)

The cointelation (sic) model allows for arbitrary short-term correlation (even $\rho = -1$) and mean reversion ($\rho = 1$) in the long term.

$$\begin{aligned} \frac{dX}{X} &= \mu dt + \sigma dW_1 && \text{leading} \\ dY &= \theta(X - Y)dy + \sigma Y dW_2 && \text{lagging} \\ d\langle W_1, W_2 \rangle &= \rho dt \end{aligned}$$

***Performance analysis
of matrix completion optimization
with applications to block causal inference***
A. Capponi and M. Stojnic (2023)

Low-rank matrix completion (nuclear norm minimization) assumes the data is missing completely at random. If the missing data shows a block pattern (e.g., corresponding to a counterfactual treatment), recovery is still possible if

$$\frac{k}{n} < 1 - 2\sqrt{\eta - \eta^2}, \quad \eta = \ell_1/n.$$

***A new entropic measure
for the causality of the financial time series***

Granger causality assumes numeric (univariate) time series. For more complex data, such as ETF transaction imbalance,

- Convert the data to images (2-dimensional fingerprints, transaction rate vs imbalance dollar value);
- Train a (convolutional) GAN to generate similar data;
- Measure if the data was easy to reproduce.

$$\text{Mean log}_2 \frac{\cos(\text{train, fake})}{\cos(\text{test, fake})} \quad \text{or} \quad \frac{\text{KL}(\text{test} \parallel \text{fake})}{\text{KL}(\text{train} \parallel \text{fake})}$$

Forking paths in financial economics
G. Coqueret (2023)

Replace bootstrap samples by “forking paths”, *i.e.*, (hundreds of) samples, obtained by varying the pre-processing options and the hyperparameters. To check for p -hacking, look at the histogram of p -values (on a linear scale): it should be (approximately) decreasing and convex. There are several forms of p -hacking:

- Simple p -hacking looks for a “path” (preprocessing pipeline) with a good p -value;
- Robust p -hacking looks for a path whose neighbourhood has good p -values, to provide robustness checks;
- Vicious p -hacking is a variant also requiring robustness to p -hacking tests.

***Estimating categorical counterfactuals
via deep twin networks***
A. Vlontzos et al. (2021)

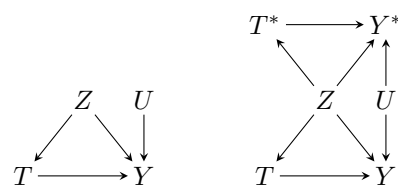
Causal inference is not limited to the study of *interventions*

$$\mathbb{E}[Y \mid \text{do}(T = 1)];$$

we can also look at *counterfactuals*

$$\mathbb{E}[Y \mid Y = 0, \text{do}(T = 1)].$$

They can be estimated with Bayesian inference on the *twin network*: duplicate T and Y ; observe T_1 and Y_1 ; intervene on T_2 ; compute the effect on Y_2 .



Unfortunately, the counterfactual is not always identifiable, e.g.,

U : unobserved, 4-valued

T : binary

$Y = X, 0, 1$ or $\neg X$, depending on U

where U has distribution $(\frac{1}{2}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6})$ or $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$. However, if $T \rightarrow Y$ is monotonic, counterfactuals are identifiable.

Train a neural network, whose architecture mimicks the twin network, with a monotonicity constraint (penalty, counter-example-guided learning, lattice network), using domain-knowledge (or average treatment effect, ATE) to order the values of T .

Metastable financial markets
D. Marcondes and A. Simonis

The market can be modeled as a hidden Markov model (HMM), e.g., with state space

$\{\text{momentum up, momentum down}\} \times \{\text{low vol, high vol}\}$.

It is *metastable*: there are clusters of states in which the market remains a very long time (there are low-probability connections between those clusters).

A changing stock-bond correlation: explaining short-term fluctuations
G. Flannery

Forecast the short-term (3-month) correlation between stocks and bonds using:

- Whether individual investors' forecasts of stock and bond markets have the same sign;
- Prospective inflation volatility;
- Whether the signs of the 21-day changes in the 2-year and 10-year government yield have been the same in the past 3 months.

Linear and non-linear causality in financial markets
H. Ma et al.

Nonlinear generalizations of Granger causality include:

- *Transfer entropy*

$$\text{TE}_{X \rightarrow Y} = H(Y_{t+1}, Y_t) + H(Y_t, X_t) - H(Y_{t+1}, Y_t, X_t) - H(Y_t)$$

(divide by $\sqrt{H(Y_{t+1}, Y_t)H(X_{t+1}, X_t)}$ to normalize to $[0, 1]$);

- Convergent cross-mapping

To separate linear and nonlinear dependence, use *Fourier transform surrogates*: start with a univariate series x , compute its Fourier transform randomize the phases, and compute the inverse Fourier transform \tilde{x} – this destroys the nonlinearities:

$$\rho_{\text{linear}}(x, y) = \rho(\tilde{x}, \tilde{y})$$

(average over several surrogates).

Some systems exhibit spans of positive and negative correlations, e.g.,

$$x_{t+1} = x_t \cdot (r_x - r_x x_t - \beta_{y \rightarrow x} y_t)$$

$$y_{t+1} = y_t \cdot (r_y - r_y y_t - \beta_{x \rightarrow y} x_t)$$

with $r_x = 3.8$, $r_y = 3.5$, $\beta_{y \rightarrow x} = 0.02$, $\beta_{x \rightarrow y} = 0.1$.

Modeling systemic risk: a time-varying nonparametric causal inference framework
J. Etesami et al.

To measure the causal relations between financial institutions, from time series data, try the “directed information graph”, which generalizes Granger causality

by looking at the regret (difference between a model with X and one without),

$$\text{DI}_{X \rightarrow Y} = \mathbb{E}_t \log \frac{P[Y_t = y_t | X_{<t}, Y_{<t}, Z_t]}{P[Y_t = y_t | Y_{<t}, Z_t]}$$

where Z are all the other variables. It is the conditional mutual information

$$E_t I(Y_t; X_{<t} | Z_{<t}, Y_{<t})$$

(use a k -nn estimator).

Analyzing stock-bond correlation: a dynamic causal system perspective
S. Du and Z. Zhang (2023)

The *convergent cross mapping* method generalizes Granger causality $X \rightarrow Y$ as $\text{cot}(Y, \hat{Y})$, where \hat{Y} is the forecast from a k -nn model (with Gaussian weights) on a delay embedding. Use $\text{CCM}_{\text{Bonds} \rightarrow \text{Equities}}$ and $\text{CCM}_{\text{Equities} \rightarrow \text{Bonds}}$ to forecast $\text{Cor}(\text{Equities}, \text{Bonds})$.

Causal network representations in factor investing
C. Howard et al. (2024)

Use DyNoTears on the constituents of the S&P 500. Applications include:

- Clustering (Node2Vec + k -means), for peer group neutralization;
- Long-short low centrality strategy;
- Market timing with the network density average eccentricity): lower eccentricity heralds left skewed and leptokurtic returns (and sometimes market crashes).

Towards automating causal discovery in financial markets and beyond
A. Sokolov et al. (2023)

Finance is not a closed, stationary system: the causal graph changes over time. LLMs can help – they only need a textual description of the variables, no actual data – the domain knowledge provided by LLMs can be used as prior for (score-based) causal discovery algorithms, but they do not scale beyond 30 nodes.

To remedy this, ask an LLM to:

- Cluster the features;
- Provide a label and a textual description for each cluster;
- Provide a causal DAG between the clusters;
- Provide a causal DAG (separately) within each cluster;
- Review those DAGs and fix any mistake made.

Then, one can take the product of the between-cluster DAG and the within-cluster DAGs.

Use do-calculus to assess the strength of each edge:

- Separately inside each cluster

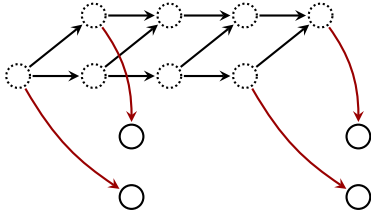
- For the between-cluster DAG, replace the features inside each cluster with their first principal component.

***On the three demons in causality in finance:
time resolution, nonstationarity
and latent factors***

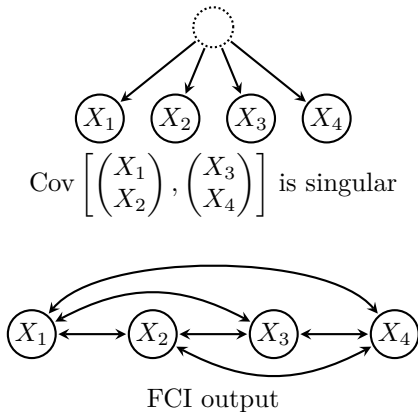
X. Dong et al.

Financial data pose a few problems for causal discovery:

- The observed data has a lower frequency than the causal data, and is often aggregated



- The data is not stationary – adding a new variable, time, can help;
- There are unobserved confounders; algorithms like FCI tend to output the most general graph (a clique for each latent variable), even when rank deficiency suggests a simpler structure.



gCastle: a Python toolbox for causal discovery

N. Zhang et al. (2021)

***Causal discovery in financial markets:
a framework for nonstationary time series data***

A. Sadeghi et al. (2024)

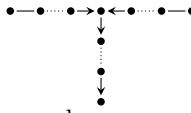
Adapt the PC algorithm, for causal discovery with non-stationary time series by:

- Adding a “time” variable;
- Considering several lags;
- Computing conditional independence tests: KCIT, RCoT, CMikm, ParCorr
- Orienting edges with known directions ($\text{Time}_t \rightarrow X_t$, $X_{t-k} \rightarrow Y_t$);
- Orienting the other edges, whenever possible (as in the original PC algorithm).

***Discovering causal models with optimization:
confounders, cycles and instrument validity***

F. Eberhardt et al.

Build a causal graph from causal independence relations by reducing the problem to a MIP (mixed integer program).

Find $x \in \{0, 1\}^{\tilde{E}}$
 $y \in \{0, 1\}^{P(\tilde{E}, \zeta)}$
 z
 Where \tilde{E} is a set of candidate edges
 $P(\tilde{E}, \zeta)$ is the set of simple paths
 $\bullet - \bullet \dots - \bullet - \bullet$ or extended paths

 Such that x and y are consistent
 y recognizes d-separation:
 $i \perp\!\!\!\perp j | C$ with error z_{ijC}
 y recognizes d-connection:
 $i \not\perp\!\!\!\perp j | C$ with error z_{ijC}
 To minimize $\sum_{ijC} z_{ijC}$

The set of candidate edges \tilde{E} is grown progressively, by adding the smallest number of edges to account for all potential colliders ($\bullet \rightarrow \bullet \leftarrow \bullet$) and non-colliders ($\bullet \rightarrow \bullet \rightarrow \bullet$ or $\bullet \leftarrow \bullet \leftarrow \bullet$) not yet explained; only include one edge for each such triple.

***Structural intervention distance (SID)
for evaluating causal graphs***

J. Peters and P. Bühlmann

Given a ground-truth causal graph g , and an estimated graph \hat{g} , the SID is the number of pairs of nodes (X, Y) for which the parents of X in \hat{g} are not a sufficient conditioning set for $X \rightarrow Y$ in g .

This is not a distance: it is not symmetric, and $\text{SID}(g, \hat{g}) = 0 \not\Rightarrow g = \hat{g}$.

The *structural Hamming distance* (SHD) between DAGs is the L^1 distance between their adjacency matrices.

***Canonical portfolios:
optimal asset and signal combination***

N. Firoozye et al. (2022)

Look for an easy-to-forecast portfolio, with CCA (canonical correlation analysis)

$$\text{Maximize}_{a,b} \text{Cor}(a'X, b'R)$$

where

X_{it} : signal
 R_{it} : return
 i : asset
 t : time.

Disciplined saddle programming
P. Schiele et al. (2023)

A convex-concave *saddle function* $f : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbf{R}$ is convex in x (for y fixed) and concave in y (for x fixed). A *saddle point* (x^*, y^*) satisfies

$$\forall x \forall y \quad f(x^*, y) \leq f(x^*, y^*) \leq f(x, y^*),$$

i.e.,

$$x^* = \underset{x}{\operatorname{Argmin}} f(x, y^*)$$

$$y^* = \underset{y}{\operatorname{Argmax}} f(x^*, y).$$

The *saddle extremum functions* are

$$G(x) = \sup_{y \in \mathcal{Y}} f(x, y)$$

$$H(y) = \inf_{x \in \mathcal{X}} f(x, y).$$

Problems with saddle point extremum constraints

$$\underset{x}{\operatorname{Minimize}} \phi(x) \text{ st } G(x) \leq 0$$

are “semi-infinite constraint” problems: there is an infinite number of constraints

$$\underset{x}{\operatorname{Minimize}} \phi(x) \text{ st } \forall y \quad f(x, y) \leq 0.$$

Robust optimization problems are saddlepoint problems.

To solve the minimax problem

$$\underset{x}{\operatorname{Minimize}} \underset{y}{\operatorname{Max}} f(x, y)$$

replace $\underset{y}{\operatorname{Max}} f(x, y)$ with its dual to have a minimin (convex optimization) problem. This can be automated for “conically representable saddle functions”; Python implementation in `dsp`.

$$f(x, y) = x'y$$

$$f(x, y) = \operatorname{convex}(x)' \operatorname{concave}(y)$$

$$f(x, y) = \left(\sum y_i x_i^2 \right)^{1/2}$$

$$f(x, y) = \log \sum y_i \exp x_i$$

$$f(x, y) = \begin{pmatrix} x \\ y \end{pmatrix}' \begin{pmatrix} P & * \\ * & -Q \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \quad P, Q \text{ psd}$$

$$f(x, Y) = x'Yx \quad Y \text{ psd}$$

Synthetic data applications in finance
V.K. Potluru et al.

To generate synthetic tabular datasets, try SDV; also look at the t-SNE scatterplot of the synthetic and real data.

Synthetic data generation is often unsupervised (agnostic to the downstream task), but it can be supervised:

- Bayesian optimization to select the model hyperparameters;

- Weights of a mixture of several data generation processes.

For event data, try Hawkes processes and their generalizations (Cox processes), and automated planning (agents have a state (accounts, balance, rent, taxes, utility bills) and a (random) goal).

For time series, try parametric models (SDE), non-parametric models (TimeGAN, QuantGAN, TimeVAE, neural SDE, implicit neural representations (hypernetworks) and agent-based models (ABIDES))

***Architectures of topological deep learning:
a survey on topological neural networks***
M. Papillon et al.

Introduction to topological deep learning (generalizations of GNNs to sets, graphs, simplicial complexes, cellular complexes, hypergraphs, combinatorial complexes), with nice diagrams.

***giotto-tda: a topological data analysis toolkit
for machine learning and data exploration***
G. Tauzin et al.

Persistent homology, Mapper algorithm, to generate features for sklearn models (with plots). Also check `scikit-tda` (easy to use), `gudhi` (rather complete), `dionysus`.

***Topological tail dependence:
evidence from forecasting realized volatility***
H.G. Souto (2023)

Compute the (birth-death) persistence diagram (PD) for one month of daily log-returns for stocks in a given basket (e.g., sector) (T days for n stocks gives T points in \mathbf{R}^n); then use the Wasserstein distance between consecutive PDs as a predictor of realized volatility.

It is an alternative to the change in average absolute correlation.

Path problems in networks
J.S. Baras and G. Theodorakopoulos (2010)

The *algebraic path problem* aggregates edges along a path, and then aggregates the results across paths:

- For the shortest path problem, we sum the edge weights along a path, and then take the path with the minimum weight;
- For the maximum reliability problem, we multiply the edge weights (probability of non-failure) along a path, and then take the path with the largest weight.

The shortest path can be computed with:

- Dijkstra (for non-negative weights);
- Bellman-Ford (if there are no negative cycles): iterate

$$d_v \leftarrow \min_u d_u + w(u, v)$$

until convergence (the updates can be asynchronous);

- Floyd-Warshall (dynamic programming, to compute the all-pairs shortest paths, by progressively increasing the set of vertices used)

$$D^0 \leftarrow A$$

$$D_{ij}^k \leftarrow \text{Min}\{D_{ij}^{k-1}, D_{ik}^{k-1} + D_{kj}^{k-1}\}.$$

The algebraic path problem on a semiring $(S, \oplus, \otimes, 0, 1)$, e.g., $(R \cup \{\infty\}, \min, +, \infty, 0)$ or $([0, 1], \max, \times, 0, 1)$, computes

$$d_{st} = \bigoplus_{p \in P_{st}} w(p)$$

$$w(p) = w(v_1, v_2) \otimes \cdots \otimes w(v_{k-1}, v_k).$$

If S is a semiring, then so are

- $S^{n \times n}$ (matrices);
- $S[[x]]$ (formal power series);
- $S[[x]]/(x^2) \simeq S \times S$;
- End S ;
- S_f (fixed points of $f : S \rightarrow S$, if f is a “reduction”).

In matrix form, the problem is $d' = d'A \oplus 1'_s$, where

d' : row vector
 A : weighted adjacency matrix
 1_s : vector of 0's, except for a 1 in position s .

Bellman-Ford iterates $d' \leftarrow d'A \oplus 1'_s$. The all-pairs shortest path problem solves $D = DA \oplus I$ (and Floyd-Warshall iterates $D \leftarrow DA \oplus I$). Under reasonable assumptions, the limit $A^* = I \oplus A \oplus A^2 \oplus \cdots$ exists; the solution is not unique but (under further reasonable assumptions) A^* is the smallest.

Applications include

- Path enumeration: algebraic path problem on S_f , where

$$S = \mathcal{P}(V \cup V^2 \cup \cdots \cup V^n)$$

\oplus : union
 \otimes : path concatenation

$$f(W) = \{w \in W : w \text{ elementary}\}$$

- Expectations, from a Markov chain (expected number of visits of a given node, expected value of a path $s \rightarrow t$);
- Minimum spanning tree;
- Shortest path;
- Widest path;
- Most reliable path;
- k shortest paths;
- etc.

Non-semiring path problems (aka non-Markovian path problems), e.g., the sortest path with discounting, can often be converted to semiring problems, by enlarging the carrier set S and selecting an appropriate reduction function f .

Neural Bellman-Ford networks: a general graph neural network framework for link prediction Z. Zhu et al. (2021)

Compute the representation of a pair of nodes (a potential edge) $g(s, t)$ from the representation of the edges on the path from s to t , with the generalized Bellman-Ford algorithm

$$f(s, t) = \bigoplus_{\substack{p: s \rightarrow t \\ p=(s=v_0, v_1, \dots, v_k=t)}} f(v_0, v_1) \otimes \cdots \otimes f(v_{k-1}, v_k)$$

where $\oplus, \otimes, 0$ are learned. Distance, Katz index, personalized page rank, etc., are special cases.

Direct preference optimization: your language model is secretly a reward model R. Rafailov et al. (2023)

RLHF (reinforcement learning with human feedback) can be reformulated without reinforcement learning.

RLHF first learns a reward function, with a Bradley-Terry model

$$P(y_1 > y_2) = \frac{e^{r(x, y_1)}}{e^{r(x, y_1)} + e^{r(x, y_2)}}$$

$$\text{loss}(\phi) = - \mathbb{E}_{x, y_1, y_2 \sim \text{Data}} \log \sigma(r_\phi(x, y_1) - r_\phi(x, y_2))$$

and uses it for RL

$$\text{Maximize}_{\pi} \mathbb{E}_{\substack{x \sim \text{Data} \\ y \sim \pi}} [r_\phi(x, y)] - \beta \cdot \text{KL}(\pi \| \pi_{\text{ref}})$$

$$\pi(y|x) = \frac{\pi_{\text{ref}}(y|x) e^{r_\phi(x, y)/\beta}}{Z(x)}.$$

But the reward can be expressed with the policy

$$r_\phi(x, y) = \beta \cdot \log \frac{\pi(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$$

and the Bradley-Terry model only uses the difference between the rewards: the partition function disappears,

$$P(y_1 > y_2|x) = \sigma \left[\beta \log \frac{\pi(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi(y_2|x)}{\pi_{\text{ref}}(y_2|x)} \right]$$

The DPO loss is

$$\text{loss}(\pi) = - \mathbb{E}_{x, y_1, y_2} \log \sigma \left[\beta \log \frac{\pi(y_1|x)}{\pi_{\text{ref}}(y_1|x)} - \beta \log \frac{\pi(y_2|x)}{\pi_{\text{ref}}(y_2|x)} \right].$$

BERTopic: neural topic modeling with a class-based TF-IDF procedure M. Grootendorst (2022)

BERTopic is a topic modeling pipeline:

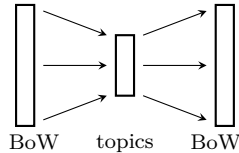
- Document embedding (sentence transformer, SBERT);
- Dimension reduction (UMAP);

- Clustering (HDBScan);
- Keyword extraction (cTF-IDF);
- Topic representation fine-tuning (either None (just use the list of keywords) or an LLM to which you provide the keywords and the documents).

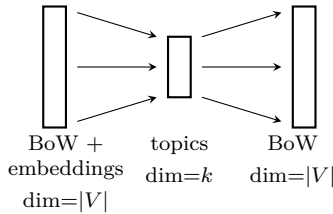
For visualization, check DataMapPlot.

PreTraining is a hot topic: contextualized document embeddings improve topic coherence
F. Bianchi et al. (2020)

ProdLDA is a VAE-based topic model.



CTM (Combined Topic Model) adds contextualized word embeddings to the input.

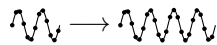


Fine-grained human feedback gives better rewards for language model training
Z. Wu et al.

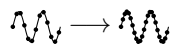
In RLHF, do not ask for a single reward for the whole answer, but one after each sentence, and decompose it into relevance, factuality and (only at the end), completeness.

Extending context window of large language models via position interpolation
S. Chen et al.

To extend the context size, do not extrapolate



but interpolate the positional embedding



Textbooks are all you need
S. Gunasekar et al.

Data quality matters and impacts the scaling laws (textbook quality web data, GPT-generated textbooks about Python coding).

Emergence of autopoietic vesicles able to grow, repair and reproduce in a minimalist particle system
T. Cabaret

Life-like particle system, with 4 types of particles: precursor, catalyst, intermediary, surfactant.

Portfolio construction when regimes are ambiguous
M. Kritzman et al.

Do not define all-or-nothing regimes

$$r_t = \mathbf{1}_{x_t \geq \text{threshold}}$$

but

$$r_t = - \underbrace{\left(\frac{x_t - x_0}{\sigma} \right)^2}_{\text{similarity}} + \underbrace{\left(\frac{x_t - \bar{x}}{\sigma} \right)^2 + \left(\frac{x_0 - \bar{x}}{\sigma} \right)^2}_{\text{informativeness}}$$

(use the Mahalanobis distance, $(x-y)'\Sigma^{-1}(x-y)$, if the indicator x is multivariate). Use the relevance r_t (truncated, to discard less relevant observations) as weights to compute regime-specific expected returns and asset covariances.

Robust statistics for portfolio construction and analysis
R.D. Martin et al.

Replace winsorization with *adaptive winsorization*: clip observations $k \cdot \text{MAD}$ away from the median; upper and lower MAD for skewed data.

M -estimators minimize

$$\sum_i \rho \left(\frac{\varepsilon_i(\theta)}{\hat{s}} \right)$$

where

- ρ is a replacement for the square loss;
- $\varepsilon_i(\theta)$ are the model residuals;
- \hat{s} is a robust scale, computed before the optimization.

Rho functions include:

- Huber (quadratic for small values, linear for large values);
- mOpt (constant for large values).

For linear regression, $\varepsilon_i(\theta) = y_i - x_i^\top \theta$ and the first order condition is

$$x_i \rho' \left(\frac{\varepsilon_i(\theta)}{\hat{s}} \right) = 0.$$

For mOpt, ρ' is

$$\psi(x) = \begin{cases} x & \text{if } |x| \leq 1 \\ \frac{\phi(1)}{\phi(1) - a} \left(x - \text{sign}(x) \frac{a}{\phi(x)} \right) & \text{if } 1 \leq |x| \leq c \\ 0 & \text{if } |x| \geq c, \end{cases}$$

with $a = 0.0132$, $c = 3.00$. It can be implemented with IRLS, with weights $w(x) = \psi(x)/x$.

The *MM estimator* computes a robust covariance by iterating

$$\begin{aligned} \sum w_i(x_i - \mu) &= 0 && \text{weighted mean} \\ \frac{1}{N} \sum w_i(x_i - \mu)(x_i - \mu)' &= C && \text{weighted covariance} \end{aligned}$$

and

$$\begin{aligned} d &= (x_i - \mu)T^\top C^{-1}(x_i - \mu) = \left\| C^{-1/2}(x_i - \mu) \right\|^2 \\ w_i &= W(d_i/cs) \end{aligned}$$

where

$$\begin{aligned} W(x) &= \begin{cases} 1 & \text{if } x \leq 4 \\ q(x) & \text{if } 4 \leq x \leq 9 \\ 0 & \text{if } x > 9 \end{cases} \\ q(x) &= \text{polynomial of degree 3} \\ s &: \text{robust scale} \\ c &: \text{tuning parameter.} \end{aligned}$$

In R, check **RobStatTM**, **PCRA** (risk analysis) and **facmodCS** (cross-sectional factor models).

Monotone operator equilibrium network **E. Winston and J.Z. Kolter**

An *operator* is a subset $F \subset \mathbf{R}^n \times \mathbf{R}^n$, seen as a multi-valued function. *Operator splitting* refers to ways of finding zeroes of a sum of operators, *i.e.*, solving $0 \in (F + G)(x)$. For a maximal monotone sum, use *forward-backward splitting*

$$x_{k+1} = R_G[x_k - \alpha F(x_k)]$$

or *Peaceman-Rachford splitting*

$$\begin{aligned} u_{k+1} &= C_F C_G(u_k) \\ x_k &= R_G(u_k). \end{aligned}$$

The *resolvent* and *Cayley* operators are

$$\begin{aligned} R_F &= (I + \alpha F)^{-1} \\ C_F &= 2R_F - I. \end{aligned}$$

If $F(x) = Gx + h$ is linear, then

$$R_F(x) = (I + \alpha G)^{-1}(x - \alpha h).$$

If $F = \partial f$ (with f convex, closed, proper (CCP)), then

$$R_F(x) = \text{prox}_f^\alpha(x) = \underset{z}{\text{Argmin}} \frac{1}{2} \|x - z\|_2^2 + \alpha f(z).$$

A *deep equilibrium network* (DEQ) model is a weight-tied, input-injected network

$$z_{i+1} = g(z_i, x).$$

A fixed point $z^* = \sigma(Wz^* + Ux + b)$ is a zero of the operator splitting problem $0 \in (F + G)(z^*)$

$$\begin{aligned} F(z) &= (I - W)z - (Ux + b) \\ G &= \partial f \\ \sigma &= \text{prox}_f^1 \\ f &\text{ CCP.} \end{aligned}$$

If $I - W \succcurlyeq mI$ for some $m > 0$, *i.e.*, if

$$W = (1 - m)I - A'A + B - B'$$

then, there is a unique solution.

Common non-linearities (ReLU, sigmoid, tanh, soft-plus) are proximal operators.

The DEQ iteration $z \leftarrow \sigma(Wz + Ux + b)$ may not converge, but the forward-backward (for small α) and the Peaceman-Rachford updates (for all α – it is also faster) do.

Monotone deep Boltzmann machines **Z. Feng et al. (2023)**

A (non-restricted) Boltzmann machine defines a joint probability distribution

$$p(x) \propto \exp \left(\sum_{ij} x'_i \Phi_{ij} x_j + \sum_i b'_i x_i \right).$$

Parametrize Φ as a block-hollow matrix

$$\begin{aligned} \Phi_{ij} &= \begin{cases} -\hat{A}'_i \hat{A}_j & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \\ \hat{A}_i &= A_i \cdot \text{Min} \left\{ \frac{\sqrt{1 - m}}{\|A_i\|_2}, 1 \right\}; \end{aligned}$$

this ensures $I - \Phi \succcurlyeq mI$. Then, the mean-field fixed point

$$q_h = \text{softmax}(\Phi_{hh}q_h + \Phi_{ho}x_o + b_h)$$

is the fixed point of a monotone deep equilibrium (mon-DEQ) model.

A non-parametric k-nearest neighbour entropy estimator **D. Lombardi and S. Pant (2015)**

To estimate the entropy from samples,

$$\hat{H} - \frac{1}{N} \sum_i \log \frac{1}{\hat{p}(x_i)},$$

the KL estimator assumes that \hat{p} is constant in $B(x_i, \varepsilon_i)$, where ε_i is the distance to the nearest neighbour

$$\hat{H} = \psi(N) - \psi(k) + \log c_d + \frac{d}{N} \sum_{i=1}^N \log \varepsilon_i$$

ψ : digamma function

$$c_d = \begin{cases} 2^d & \text{for } \|\cdot\|_\infty \\ \frac{\pi^{d/2}}{\Gamma(1 + d/2)} & \text{for } \|\cdot\|_2 \end{cases}$$

In particular, this assumes the distribution is isotropic in the ball $B(x_i, \varepsilon_i)$. Instead, the KpN estimator assumes it is a (truncated) Gaussian $N(\mu_i, \Sigma_i)$, where μ_i, Σ_i are the sample mean and variance of the $p \geq k$ nearest neighbours of x_i :

$$g_i(x) = \exp -\frac{1}{2}(x - \mu_i)' \Sigma_i^{-1} (x - \mu_i)$$

$$G_i = \int_{B(x_i, \varepsilon_i)} g_i(x) dx$$

$$\hat{H} = \psi(N) - \psi(k) - \frac{1}{N} \sum \log \frac{g_i(x_i)}{G_i}.$$

Computing the Gaussian integral G_i is tricky, but can be done, for the L^∞ ball, with expectation propagation multivariate Gaussian probability.

***Gaussian probabilities
and expectation propagation***
J.P. Cunningham et al. (2011)

How to compute a Gaussian integral

$$\mathbb{P}_{X \sim N(\mu, \Sigma)} \left[X \in \prod_i [\ell_i, u_i] \right]$$

approximately, with expectation propagation.

Flow annealed importance sampling bootstrap
L.I. Midgley et al.

Importance sampling (IS) samples from a distribution p_0 and reweights the samples to match a distribution p_1 .

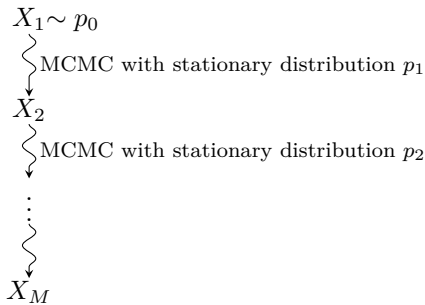
$$x \sim p_0$$

$$w(x) = \frac{p_1(x)}{p_0(x)}$$

Annealed importance sampling (AIS) uses a chain of distributions,

$$\log p_i = \beta_i \log p_0 + (1 - \beta_i) \log p_M,$$

progressively moving from p_0 to p_M ,



$$w(x_M) = \frac{p_1(x_1) p_2(x_2) \dots p_M(x_M)}{p_0(x_1) p_1(x_2) \dots p_{M-1}(x_M)}$$

Application to normalizing flows and the α -divergence, for $\alpha = 2$ (to limit mode collapse)

$$D_\alpha(p||q) = \frac{1}{\alpha(1-\alpha)} \int p(x)^\alpha q(x)^{1-\alpha} dx$$

$$D_0(p||q) = \text{KL}(q||p)$$

$$D_1(p||q) = \text{KL}(p||q)$$

Conditional random fields: an introduction
H.M. Wallach (2004)

A *conditional random field* (CRF) is a conditional probability distribution on sequences of labels, often used to tag text (e.g., POS tagging, NER, etc.), of the form

$$P(Y|X) = \frac{1}{Z(X)} \sum_j \lambda_j \underbrace{\sum_i f_j(y_{i-1}, y_i, x, i)}_{F_j(y, x)}$$

where the f_j are binary features of y_i , or y_i and y_{i-1} , possibly involving x . It is a log-linear probabilistic graphical model.

The log-likelihood is

$$\ell(\lambda) = \sum_{(x, y) \sim \text{data}} -\log Z(x) + \sum_j \lambda_j F_j(y, x).$$

Its gradient is

$$\frac{\partial \ell}{\partial \lambda_j} = \mathbb{E}_{(x, y) \sim \text{data}} F_j(y, x) - \sum_{x \sim \text{data}} \mathbb{E}_{y \sim p(y|x, \lambda)} F_j(y, x).$$

The normalization factor can be computed with a matrix product

$$Z(x) = [\prod M_i(x)]_{\text{start, end}}$$

where

$$M_i(x)_{y', y} = \exp \sum_j \lambda_j (y', y, x, i)$$

after adding labels $y_0 = \text{start}$, $y_{n+1} = \text{end}$.

The second term in the gradient can be computed with dynamic programming (forward-backward algorithm)

$$\mathbb{E}_{y \sim p(y|x, \lambda)} F_j(y, x)$$

$$= \sum_y p(y|x, \lambda) F_j(y, x)$$

$$= \sum_i \sum_{y, y'} p(Y_{i-1} = y', Y_i = u|x, \lambda) f_j(y', y, x)$$

$$\alpha_0(y|x) = \mathbf{1}_{y=\text{start}} \quad \alpha_i(x) = M_i(x)' \alpha_{i-1}(x)$$

$$\beta_{n+1}(y|x) = \mathbf{1}_{y=\text{end}} \quad \beta_i(x) = M_{i+1}(x) \beta_{i+1}(x)$$

$$P(Y_{i-1} = y', Y_i = y|x, \lambda) = \frac{\alpha_{i-1}(y'|x) M_i(y', y|x) \beta_i(y|x)}{Z(x)}$$

Implicit MLE: backpropagating through discrete exponential family distributions
M. Niepert (2021)

To train a model involving sampling from discrete random variables

$$x \xrightarrow{h_V} \theta \mapsto z \sim p_\theta(z) \xrightarrow{f_U} y$$

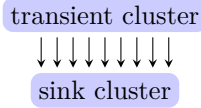
$$\text{loss}(u, v) = \mathbb{E}_{z \sim p_{n_v}(x)} [\ell(f_u(z), y)]$$

implicit MLE uses

$$\begin{aligned} \text{forward:} \quad & \varepsilon \sim \rho \\ & \hat{z} = \text{MAP}(\theta + \varepsilon) \\ \text{backward:} \quad & \theta' = \theta - \lambda \nabla_z \ell \\ & \hat{\nabla}_\theta \ell = \hat{z} - \text{MAP}(\theta' + \varepsilon). \end{aligned}$$

DIGRAC: Digraph clustering based on flow imbalance
Y. He et al.

DMPA finds clusters in a directed graph, assuming that the directionality contains relevant information: identify the clusters from the imbalance between them.



n : number of nodes

K : number of clusters

$P \in \mathbf{R}^{n \times k}$: cluster membership

$W_{k\ell} = P'_{\cdot k} A P_{\cdot \ell}$: probabilistic cut

$|W_{k\ell} - W_{\ell k}|$: imbalance flow

$V_k = (\mathbf{1}'(A + A')P)_k$: volume

$$C_{k\ell} = 2 \frac{|W_{k\ell} - W_{\ell k}|}{V_k + V_\ell} \in [0, 1]$$

loss = Mean $C_{k\ell}$
 $k < \ell$

or loss = Top- β Mean $C_{k\ell}$ (with $\beta = 1$ or 3)
 $k < \ell$

Use a GNN to compute P .

MagNet: a neural network for directed graphs
X. Zhang et al.

Graph convolution assumes the graph is undirected. For directed graphs, consider the magnetic Laplacian instead (a complex Hermitian matrix, which depends

on a parameter q).

$$A_s = \frac{1}{2}(A + A')$$

$$D_s = \text{diag}(A_s \mathbf{1})$$

$$\Theta^{(q)} = 2\pi q(A - A')$$

$$H^{(q)} = A_s \odot \exp(i\Theta^{(q)}) \text{ (element-wise)}$$

$$L^{(q)} = D_s - H^{(q)}$$

$$\tilde{L}^{(q)} = I - (D_s^{-1/2} A_s D_s^{-1/2}) \odot \exp(i\Theta^{(q)})$$

Statistical inference on random dot product graphs: a survey
A. Athreya et al. (2017)

The adjacency matrix of an *independent-edge random graph* is

$$A_{ij} \sim \text{Bernoulli}(P_{ij})$$

where $P \in [0, 1]^{n \times n}$.

For a *latent position* random graph, $p_{ij} = \kappa(x_i, x_j)$, $x_i \in \mathbf{R}^d$, $\kappa : \mathbf{R}^d \times \mathbf{R}^d \rightarrow [0, 1]$.

For a *random dot product* graph, $\kappa(x, y) = x'y$, i.e., $P = XX'$.

For a *stochastic block model* (SBM), X has repeated rows.

The Erdős-Rényi graph is even simpler: $p_{ij} = p$ is constant.

The latent positions can be recovered by clustering the *adjacency spectral embedding*.

$$\hat{X} = U_A S_A^{1/2}$$

$$|A| = (A'A)^{1/2}$$

$$|A| \approx U_A S_A U_A' \text{ truncated spectral decompositions}$$

or the *Laplacian spectral embedding*

$$\check{X} = U_L S_L^{1/2}$$

$$|L| \approx U_L S_L U_L'$$

Digraph inception convolutional networks
Z. Tong et al. (2020)

Spectral graph convolution (GCN) assumes the graph is undirected.

$$X \mapsto \hat{A} X \Theta$$

$$\tilde{A} = A + I$$

$$\tilde{D} = \text{diag}(\tilde{A} \mathbf{1})$$

$$\hat{A} = \tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}$$

For a directed graph, the page-rank Laplacian is dense, because of the teleportation to arbitrary nodes.

$$P_{\text{rw}} = D^{-1} A$$

$$P_{\text{pr}} = (1 - \alpha) P_{\text{rw}} + \frac{\alpha}{n} \mathbf{1}_{n \times n}$$

π_{pr} : stationary distribution for P_{pr} (unique if $\alpha \in (0, 1)$)

$$\Pi_{\text{pr}} = \text{diag}(\pi_{\text{pr}})$$

$$P_{\text{pr}} = I - \frac{1}{2} [\Pi^{1/2} P \Pi^{-1/2} + \Pi^{-1/2} P' \Pi^{1/2}]$$

Instead, add a node, connected to all other nodes, and only teleport to it (*i.e.*, consider its personalized page rank)

$$P_{\text{ppr}} = \begin{pmatrix} (1-\alpha)\tilde{D}^{-1}\tilde{A} & \alpha\mathbf{1} \\ \frac{1}{n}\mathbf{1}' & 0 \end{pmatrix}.$$

Multiscale directed graph convolution uses the k th order proximity matrix

$$\begin{aligned} P_0 &= I \\ P_1 &= \tilde{D}^{-1}\tilde{A} \\ P_{k+1} &= (P_1)^k (P_1')^k \cap (P_1'')^k (P_1)^k. \end{aligned}$$

Stability of China's stock market: measure and forecast by Ricci curvature on network
X. Wang et al. (2022)

The average curvature (Ollivier, Forman, Menget or Haantjes – compute all 4) of the truncated graph ($\rho = 0.75$) of the correlation matrix of daily stock returns (on a 1-month window) is a measure of market stability.

Graph neural networks are dynamic programmers
A. Dudzik and P. Veličković (2022)

Dynamic programming, *i.e.*, algorithms of the form

$$d[x] \leftarrow \text{Aggregate}_{y \in \text{Expand}(x)} [\text{score}(d[y], d[x])]$$

are very similar to GNNs. This similarity can be formalized.

$$\begin{array}{ccc} X \xrightarrow{p} Y & & [X, R] \xrightarrow{p \otimes} [Y, \text{list}(R)] \xrightarrow{\otimes} [Y, R] \\ i \downarrow & \searrow \scriptstyle (R, \otimes, \oplus) & \downarrow \scriptstyle o \oplus \\ W & & Z \\ & \nearrow \scriptstyle i^* & [Z, \text{bag}(R)] \\ & & \downarrow \oplus \\ & & [W, R] & & [Z, R] \end{array}$$

Exact combinatorial optimization with graph convolutional neural networks
M. Gasse et al. (2019)

Imitation learning of branch-and-bound selection policies, with a GCN on the variable-constraint bipartite graph.

Neural combinatorial optimization with reinforcement learning
I. Bello et al. (2017)

A *pointer network* can generate a permutation $\sigma \in \mathfrak{S}_n$ of a set of points x_1, \dots, x_n :

- First, an encoder processes the points in sequence, with an LSTM;
- Then, a decoder generates the $\sigma(i)$'s, one at a time: at each step, it generates a distribution on the x_i 's (attention), picks a point from that distribution, and gives it to the next step.

Solve the TSP with RL (A3C, with policy gradient, and REINFORCE to compute the gradient), with a pointer network for the policy (actor).

Graph neural networks inspired by classical algorithms
Y. Yang et al.

The energy

$$\ell(Y) = \|Y - f_w(X)\|^2 + \lambda \text{tr}(Y'LY)$$

where Y are candidate node embeddings, X are node features, and $L = D - A$ is the Laplacian, is minimized by

$$Y^*(w) = (I + \lambda L)^{-1} f_w(X);$$

it is an approximation of $f_w(X)$ smoothed by the graph structure.

It can be used for node classification

$$\text{loss}(\theta, w) = \sum_i D[g_\theta(y_i^*(w)), t_i].$$

Instead of inverting $I + \lambda L$, do a few gradient steps on ℓ ; they can be interpreted as GNN layers.

The penalty $\text{tr}(Y'LY) = \sum_{ij \in E} \|y_i - y_j\|^2$ can be generalized to $\sum_{ij \in E} \rho(\|y_i - y_j\|^2)$ to increase robustness (to adversarial edges).

On the ability of graph neural networks to model interactions between vertices
N. Razin et al.

The *walk index* of a graph (V, E) wrt a partition $V = I \cup I^c$ is the number of length- L walks starting on the boundary of the partition. The *walk index edge sparsification* algorithm

- Starts with a set of partitions across which we want to keep interactions
- Computes the walk indices wrt each of those partitions if we remove one edge, for each edge;
- Removes the edge with the maximum walk indices (aggregated with max, min or mean).



The disruption index is biased by citation inflation
A.M. Petersen et al.

A disruptive node, in a citation network, looks like



(widely cited, but the papers it cites are no longer widely cited).

***Training spiking neural networks
using lessons from deep learning***
J.K. Eshraghian et al.

The leaky integrate-and-fire (LIF) neuron takes as input a spike train , smooths it into a “potential”, e.g., with exponential decay, , and, each time it exceeds a prescribed threshold, emits a spike, each emission resetting the potential.

$$\begin{aligned} X_t &: \text{input} \\ U_t &= \beta U_{t-1} + W x_t - \theta S_{t-1} \text{ latent state} \\ S_t &= \mathbf{1}_{U_t \geq 0} \text{ output} \end{aligned}$$

The input needs to be converted into a spike train, e.g., as the firing rate (rate coding), or the time to the first spike (latency coding).

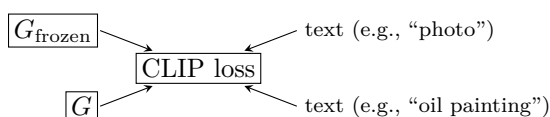
Spiking neural nets (SSN) can be trained with

- Shadow training (SSN in the forward pass, ANN in the backward pass);
- Back-propagation through time (BPTT);
- Local learning rules.

***SpykeTorch: Efficient simulation
of convolutional spiking neural networks
with at most one spike per neuron***
M. Mozafari et al. (2019)

In unsupervised learning with STDP (spike timing dependent plasticity), increase $w_{j \rightarrow i}$ if j fires before i . For supervised learning, use STDP or anti-STDP depending on whether the output is correct.

***StyleGAN-NADA: CLIP-guided
domain adaptation of image generators***
R. Gal et al.



***Brax: a differentiable physics engine
for large scale rigid body simulation***
C.D. Freeman et al.

JAX alternative to MuJoCo.

***Gotta go fast when generating data
with score-based models***
A. Jolicœur-Martineau et al.

Better SDE solver for denoising diffusions

***Agent57:
outperforming the Atari human benchmark***
A.P. Badia et al.

NGU (never give up) provides intrinsic motivation by augmenting the reward to promote novelty, both within and between episodes.

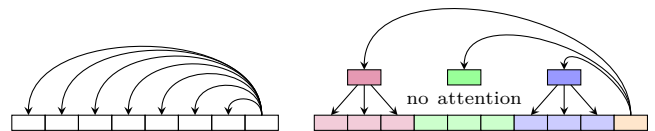
Agent57 does not learn a single policy, but a family of policies, from very exploratory to purely exploitative.

***STaR: self-taught reasoner
Bootstrapping reasoning with reasoning***
A. Zelikman et al.

To turn few-shot learning into many-shot learning, give a few examples to the LLM, ask it to generate more, prune those leading to incorrect answers, and use the rest for fine-tuning.

***Towards mental time travel: a hierarchical
memory for reinforcement learning agents***
A.K. Lampinen et al. (2021)

To help RL agents remember, store the past in chunks, perform attention on chunk summaries, then attention on relevant chunks (hierarchical attention).



***Training data-efficient image transformers
and distillation through attention***
H. Touvron et al.

Vision transformers add a (learned) class token to the patch tokens (not unlike the [CLS] token for language) and use it to forecast the true label.

“Distillation through attention” adds another “distillation” token, and uses it to forecast the teacher’s hard label (rather than the teacher’s softmax).

***An optimal control perspective
on diffusion based generative modeling***
J. Berner et al. (2022)

The marginals of a time-reversed Itô process can be represented as the marginals of another Itô process. If

$$\begin{aligned} Y_0 &\sim \mathcal{D} \\ dY &= f(Y, s)ds + \sigma(s)dB_s \\ X_0 &\sim Y_T \\ dX &= \mu(X, s)ds + \sigma(s)dB_s \\ \mu &= \sigma\sigma' \nabla_x \log p_Y - f \\ f &: \mathbf{R}^d \times [0, T] \longrightarrow \mathbf{R}^d \\ \sigma &: [0, T] \longrightarrow \mathbf{R}^{d \times d}, \end{aligned}$$

then $X_T \sim \mathcal{D}$. More generally, $p_T = \tilde{p}_X$ a.e., i.e., $Y_{T-t} \stackrel{d}{=} X_t$ a.e.

***Language models
(mostly) know what they know***
S. Kadavath et al.

Their self-evaluation is better calibrated if they are provided with several of their own samples.

**Non-adversarial training of neural SDEs
with signature kernel scores**
Z. Issa et al.

Find θ such that the SDE

$$dX_t = \mu_\theta(X_t, t)dt + \sigma_\theta(X_t, t)dW_t$$

minimizes

$$\mathcal{L}(\theta) = \mathbb{E}_{Y \sim \text{data}} \left[\mathbb{E}_{x, x' \sim p_\theta} k(x, x') - 2 \mathbb{E}_{x \sim p_\theta} k(x, y) \right]$$

where the *signature kernel* is defined by a PDE

$$\begin{aligned} k(x, y) &= \langle S(x), S(y) \rangle = f(T, T) \\ f(s, t) &= \int_0^t \int_0^s f(u, v) \langle dx_u, dx_v \rangle. \end{aligned}$$

**Higher order kernel mean embeddings
to capture filtrations of stochastic processes**
C. Salvi et al.

The signature kernel is

$$k(x, y) = \langle S(x), S(y) \rangle$$

where x, y are paths $[0, T] \rightarrow \mathbf{R}^d$ and S is the signature transform. It can be computed as $k(x, y) = u(T, T)$, where u is the solution of

$$\begin{aligned} \frac{\partial^2 u}{\partial s \partial t} &= \langle \dot{x}(s), \dot{y}(t) \rangle u \\ u(0, \cdot) &= u(\cdot, 0) = 1 \end{aligned}$$

**Deep into the domain shift: transfer learning
through dependence regularization**
S. Ma et al.

Domain adaptation learns domain-invariant features.

Separately penalize differences in the dependence structure (copula) and in the marginals.

**Sophia: a scalable stochastic second-order
optimizer for language model pretraining**
H. Liu et al.

Use the diagonal Hessian as preconditioner; only update it every few iterations; clip the updates.

Estimate the diagonal Hessian as (Hutchinson, unbiased)

$$\begin{aligned} H &\approx u \odot \nabla \langle \text{loss}, u \rangle \\ u &\sim N(0, I) \end{aligned}$$

or (Gauss-Newton-Bartlett, biased)

$$\begin{aligned} \hat{g} &= \nabla \frac{1}{B} \sum_b \text{loss}(f_\theta(x_b), \hat{y}_b) \\ \hat{y}_b &\sim \text{softmax } f_\theta(x_b). \end{aligned}$$

Transformers can do Bayesian inference
S. Müller et al.

A Prior data fitted network (PFN) forecasts the (posterior) distribution

$$q_\theta : y_{n+1} \mid (x_1, y_1), \dots, (x_n, y_n), x_{n+1}.$$

The distribution q_θ is modeled as a “Riemann distribution”: a discretized continuous distribution with equal (prior) probability bins (if the marginal distribution of y is not bounded, use a half-Gaussian for the tails). Use a transformer encoder (with no positional encoding): it is \mathfrak{S}_n -invariant.

Applications include sampling from Gaussian processes (with a hyper-prior), tabular data, Bayesian neural nets.

Guided generation of cause and effect
Z. Li et al.

Train a model which, given a sentence, generates possible causes and consequences; to ensure diversity, constrain them to contain words from the CEG.

Datasets:

- Causal bank: cause-effect pairs, obtained from Common Crawl by pattern matching;
- Cause-effect graph (CEG): directed graph, with words as nodes, and links when they appear on either side of a causal statement.

**Span-based joint entity and relation extraction
with transformer pretraining**
M. Eberts and A. Ulges (2021)

SpERT uses fine-tuned BERT embeddings to filter and classify spans (NER) and relations

$$[i, i+k] \mapsto f(e_i, \dots, e_{i+k}) \sqcup w_{k+1} \sqcup c$$

where $[i, i+k]$ is the candidate span, e_i, \dots, e_{i+k} are the token embeddings, $k+1$ is the span length, w_{k+1} is a learned embedding of the span length, c is the embedding of the CLS token (of the whole sentence), \sqcup is concatenation.

For relations, use the max-pooling of the spans and of the context.

**SpanBERT: improving pre-training
by representing and predicting spans**
M. Joshi et al.

BERT variant which masks spans rather than isolated tokens.

**Causal inference using invariant prediction:
identification and confidence intervals**
J. Peters et al. (2015)

Causal models are robust against interventions: in particular, a causal model has the same predictive accuracy under different experimental settings or regimes

(i.e., various interventions – we do not need to know what those interventions are).

gCastle: a Python toolbox for causal discovery
K. Zhang et al.

GRAPE for fast and scalable graph processing and random-walk-based embedding
L. Cappelletti et al. (2021)

Python library for graph features (like `networkx`, `igraph`, `networkkit`, `graph_tool`), random-walk-based node embeddings, and edge prediction.

An empirical evaluation of time series feature sets
T. Henderson and B.D. Fulcher

In Python, check `catch22`, `tsfresh` (raw Fourier coefficients: real and imaginary parts, angle), `kats`, `tsfel`. In R, check `catch22`, `feats`, `tsfeatures`. Compare them on the “1000 empirical time series” dataset.

Multilevel nested simulation for efficient risk estimation
M.B. Giles et al.

To compute $E[H(E[X|Y])]$, where H is the Heaviside function, we can use a coarse Monte Carlo approximation of $E[X|Y]$ if it is far away from zero, and a finer one if it is closer.

Portfolio optimization rules beyond the mean-variance approach
M. Markov and V. Markov (2023)

The asymmetric Laplace distribution is

$$p_{\mu, \sigma_+, \sigma_-}(x) \propto \exp - \frac{|x - \mu|}{\sigma_{\text{sign}(x - \mu)}}.$$

Linear time GPs for inferring latent trajectories from neural spike trains
M. Dowling et al. (2023)

The Hida-Matérn kernel is

$$\kappa(\tau) = \cos(b\tau) \kappa_{\text{Matérn}}(\tau, \nu + \tfrac{1}{2}).$$

Large non-stationary noisy covariance matrices: a cross-validation approach
V.W.C. Tan and S. Zohren (2020)

The sample covariance matrix with exponential

weights

$$\begin{aligned} \hat{E} &= \frac{\sum_{t=0}^{T-1} \beta^t x_{\tau-t} x'_{\tau-t}}{\sum \beta^t} \\ &= \frac{1}{T} C' B X \\ B &= T \text{diag} \left(\frac{\beta^t}{\sum \beta^s} \right)_t \\ &= T \frac{1 - \beta}{1 - \beta^T} \text{diag}(\beta^t)_t \\ \hat{E} &= \text{Cov} \tilde{X} \\ \tilde{X} &= X B^{1/2} \\ N/T &\longrightarrow q \\ N(1 - \beta) &\longrightarrow q_e \end{aligned}$$

can account for non-stationarity, but its limiting spectral distribution is slightly different from the Marchenko-Pastur one: the eigenvalues of \hat{E} are biased estimators of the true eigenvalues. Keep the eigenvectors, but use cross-validation (on \tilde{X}) to remove the bias of the eigenvalues.

Let ρ be the asymptotic distribution of the eigenvalues of a random matrix; its *Stieltjes transform* is

$$G(z) = \int_{\mathbf{R}} \frac{\rho(t)}{z - t} dt = \lim_{N \rightarrow \infty} \frac{1}{N} \text{E tr}[(X - zI)^{-1}].$$

The *R-transform* is defined by

$$R(G(z)) + \frac{1}{G(z)} = z$$

and satisfies $R_{A+B} = R_A + R_B$ (assuming the eigenvectors are in “general position”). To compute the spectrum of $A + B$:

- Compute G_A, G_B, R_A, R_B ;
- Compute $R_{A+B} = R_A + R_B$;
- Compute G_{A+B} ;
- Compute ρ_{A+B}

Data-driven distributionally robust risk parity portfolio optimization
G. Costa and R.H. Kwon (2021)

To estimate expected returns and variance matrix from weighted historical data, with weights in a user-specified ambiguity set, alternate between:

- Gradient ascent, to find adversarial weights;
- Risk parity minimization.

Portfolio optimization using predictive auxiliary classifier generative adversarial networks with measuring uncertainty
J. Kim and M. Lee

GANs can be used as predictive models: they do not output a single value, but (samples from) a distribution, accounting for prediction uncertainty.

**Multi-modal deep learning
for credit rating prediction
using text and numerical data streams**
M. Tavakoli et al.

Cross-attention for multimodal learning, viz, predicting credit ratings from structured (financial ratios) and unstructured (earnings calls transcripts) data.

The virtue of complexity in return prediction
B. Kelly et al. (2022)

Simple models understate return predictability: do not shy away from complex ML models.

**Embedding and correlation tensor for XRP
transaction networks**
A. Chakraborty et al. (2022)

Take XRP transactions:

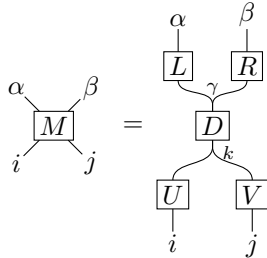
- For each week, compute a network, with wallets as nodes and transactions as edges;
- Compute node embeddings (deepwalk or node2vec);
- Only keep nodes present each week;
- Compute the correlations

$$M_{ij}^{\alpha\beta} = \text{Cor}_{[t-\Delta T, t+\Delta T]}[V_i^\alpha, V_j^\beta]$$

where i, j are nodes, and α, β coordinates.

- Compute a double SVD

$$\begin{aligned} M_{ij}^{\alpha\beta} &= U_{ik} \Delta_k^{\alpha\beta} V_{kj} \\ \Delta_k^{\alpha\beta} &= L^{\alpha\gamma} D_k^\gamma R^{\gamma\beta} \\ \text{i.e., } M_{ij}^{\alpha\beta} &= D_k^\gamma (U_{ik} V_{kj}) (L^{\alpha\gamma} R^{\gamma\beta}) \end{aligned}$$



**Generative AI in active management:
QesGFT**
Wolfe Research

Model trained on a point-in-time corpus (Wikipedia, books, news, policy documents (UN, OECD, etc.), preprints, regularoty filings, conference calls, corporate websites), from GPT-3, with self-instruct, flash attention (GPU tiling), 8-bit quantization, RoPE.

**A graph theory approach
to portfolio optimization**
D. Cajas (2023)

From the correlation matrix, build the MST or the TMFG; add a constraint to the optimization problem, to invest (on average) in low-centrality assets

$$w' \cdot \text{centrality} = c.$$

To increase diversity further, do not invest in nearby assets: this can be formulated as a MIP, or relaxed to a SDP: $B \odot (ww') = 0$, where B_{ij} indicates if there is a path of length at most ℓ between i and j , is a quadratic non-convex constraint, but ww' can be replaced by a positive semidefinite approximation X

$$ww' = \underset{X: \begin{bmatrix} X & w \\ w' & 1 \end{bmatrix} \succcurlyeq 0}{\text{Argmin}} \text{tr } X$$

by adding a $\lambda \text{tr } X$ penalty to the objective.

**Post-covid inflation and the monetary policy
dilemma: an agent-based scenario analysis**
M.S. Knicket et al.

The *Mark-0 agent-based model* is a macro-economic model accounting for savings, wages, energy sector, interest, inflation expectation, consumption, non-matched demand and production, several goods, several firms, price, target production.

Maximally machine-learnable portfolios
P.G. Coulombe et al. (2022)

ACE (alternating conditional expectation) finds and easy-to-forecast transformation of the target variable

$$g(Y) = f(X) + \varepsilon$$

(iteratively estimate g fixing f , and f fixing g , with backfitting polynomials). With time series, this becomes

$$g(Y_{t+h}) = f(X_t) + \varepsilon_{t+h}.$$

This can be generalized to portfolios (MACE, multi-variate ACE): find a portfolio w whose returns

$$g(Y_{t+h}) = w'Y_{t+h}$$

are easy to forecast with a random forest (from market variables, macro variables, or the characteristics (returns, earnings yield, etc.) of the portfolio itself).

$$\begin{aligned} &\underset{w, f}{\text{Minimize}} \sum_t [w' r_{t+h} - f(X_t)] + \lambda \|w\|^2 \\ &\text{such that } w \geq 0, w' \mathbf{1} = 1. \end{aligned}$$

The model can be estimated by alternatingly fitting a random forest to get f , and a ridge regression to get w .

**Comparing deep learning models for volatility
prediction using multivariate data**
W. Ge et al.

Temporal fusion transformers (TFT) and temporal convolution networks (TCN) outperform GARCH-like models.

Temporal fusion transformers for interpretable multi-horizon time series forecasting
B. Lim et al.

Stack:

- LSTM
- GRN (gated residual network)

$$\text{GLU} = \sigma((W_1x + b_1) \odot (W_2x + b_2))$$

- Multi-head attention;
- GRN

to forecast several quantities, at several horizons.

GFlowNet Foundations
Y. Bengio et al.

Contrary to MCMC, GFlowNets can efficiently sample from multimodal distributions, on discrete objects that can be constructed incrementally, e.g., set, graphs or molecules. They can be trained, not from data, but from an unnormalized probability distribution (the *reward*).

A *flow network* is a DAG, with a single source and a single sink, with a non-negative function $F : \mathcal{T} \rightarrow \mathbf{R}_+$ on the set of complete trajectories. This defines:

- State flow: $F(s) = \sum_{\tau : s \in \mathcal{T}} F(\tau)$;
- Edge flow: $F(s \rightarrow s') = \sum_{\tau : (s \rightarrow s') \in \mathcal{T}} F(\tau)$;
- Total flow: $Z = \sum_{\tau} F(\tau)$;
- Forward transition probability:

$$P_F(s' | s) = \frac{F(s \rightarrow s')}{F(s)};$$

- Backward transition probability:

$$P_B(s | s') = \frac{F(s \rightarrow s')}{F(s')}.$$

State and edge flows define a flow if they satisfy the *flow matching conditions*

$$\begin{aligned} \hat{F}(s') &= \sum_{s \rightarrow s'} \hat{F}(s \rightarrow s') \\ \hat{F}(s') &= \sum_{s' \rightarrow s''} \hat{F}(s' \rightarrow s''). \end{aligned}$$

State flow and forward and backward transition probabilities correspond to a flow if they satisfy the *detailed balance conditions*

$$\hat{F}(s) \hat{P}_G(s' | s) = \hat{F}(s') \hat{P}_B(s | s').$$

A flow is *Markovian* if

$$P(s \rightarrow s' | s_0 \rightarrow \dots \rightarrow s) = P(s \rightarrow s' | x) = P_F(s' | s).$$

Each flow function (on paths) is equivalent to (has the same edge flow as) a unique Markovian flow. The edge flows determine a unique Markovian flow.

A GFlowNet parametrized by edge flows $\hat{F}(s \rightarrow s')$ can be trained with the *flow matching loss*

$$\sum_{s'} \left[\log \frac{\delta + \sum_{s \in \text{Par}(s')} \hat{F}(s \rightarrow s')}{\delta + R(s') + \sum_{\substack{s'' \in \text{Child}(s') \\ s'' \neq \text{final}}} \hat{F}(s' \rightarrow s'')} \right]^2$$

A GFlowNet parametrized by vertex flows $\hat{F}(s)$ and transition probabilities $\hat{P}_F(s' | s)$, $\hat{P}_B(s | s')$ can be trained with the *detailed balance loss*

$$\sum_{s \rightarrow s'} \left[\log \frac{\delta + \hat{F}(s) \hat{P}_F(s' | s)}{\delta + \hat{F}(s') \hat{P}_B(s | s')} \right]^2$$

(replace the denominator with $\delta + R(s)$ if $s' = \text{final}$).

A GFlowNet parametrized by total flow \hat{Z} and transition probabilities $\hat{P}_F(s' | s)$, $\hat{P}_B(s | s')$ can be trained with the *trajectory balance loss* (preferred)

$$\sum_{\tau} \left[\log \frac{\hat{Z} \prod_t \hat{P}_F(s_t | s_{t-1})}{R(s_T) \prod_t \hat{P}_B(s_{t-1} | s_t)} \right]^2.$$

Flow network based generative models for non-iterative diverse candidate generation
E. Beggio et al.

Original GFlowNet paper, with applications to molecule synthesis, and a comparison with MCMC and PPO.

Trajectory balance: improved credit assignment in GFlowNets
N. Malkin et al.

GFlowOut: dropout with generative flow networks
D. Liu et al. (2023)

Learn the distribution of the dropout binary masks, in a sample-dependent or sample-independent way, with a GFlowNet.

Deep neural networks with dropout perform approximate Bayesian inference and approximate the posterior of a deep Gaussian network.

Direct parametrization of Lipschitz-bounded deep networks
R. Wang and I.R. Manchester (2023)

γ -Lipschitz neural nets

$$\begin{aligned} z_0 &= x \\ z_{k+1} &= \sigma(W_k z_k + b_k) \\ y &= W_L z_L + b_L \end{aligned}$$

can be parametrized as

$$\begin{aligned} b_k &\in \mathbf{R}^{n_{k+1}} \\ d_j &\in \mathbf{R}^{n_j} \\ X_k &\in \mathbf{R}^{n_{k+1} \times n_{k+1}} \\ Y_k &\in \mathbf{R}^{n_k \times n_{k+1}} \\ \Psi_k &= \text{diag}(e^{d_k}) \\ \begin{bmatrix} A'_k \\ B_k \end{bmatrix} &= \text{Cayley} \begin{bmatrix} X_k \\ Y_k \end{bmatrix} \\ A_{-1} &= I \\ \Psi_{-1} &= \sqrt{\frac{\gamma}{2}} I \\ W_k &= 2\Psi_{k-1}^{-1} B_k A'_{k-1} \Psi_{k-1} \end{aligned}$$

where the Cayley transform is

$$\begin{aligned} \text{Cayley} \begin{bmatrix} X \\ Y \end{bmatrix} &= \begin{bmatrix} (I + Z)^{-1}(I - Z) \\ -2Y(I - Z)^{-1} \end{bmatrix} \\ Z &= X - X' + Y'Y \end{aligned}$$

Equivalently, the model can be written with *Lipschitz sandwich layers*

$$\begin{aligned} h_0 &= \sqrt{\gamma}x \\ h_{k+1} &= \sqrt{2}A'_k \Psi_k \sigma(\sqrt{2}\Psi_k^{-1} B_k h_k + b_k) \\ y &= \sqrt{\gamma}B_L h_L + b_L, \end{aligned}$$

**Recurrent equilibrium networks:
flexible dynamic models
with guaranteed stability and robustness**
M. Revay et al.

A recurrent equilibrium network (REN) is a state space model

$$\begin{aligned} x_{t+1} &= Ax_t + B_1 w_t + B_2 u_t + b_x \\ y_t &= C_2 x_t + D_{21} w_t + D_{22} u_t + b_y \end{aligned}$$

whose weights are given by an equilibrium network

$$w_t = \sigma(D_{11}w_t + C_1 x_t + D_{12}u_t + b_v).$$

Sufficient conditions for the model to be Lipschitz can be translated into direct parametrizations.

A cookbook of self-supervised learning
R. Balestrierio et al.

There are 4 families of self-supervised learning (SSL) models:

- Contrastive loss (SimCLR);
- Self-distillation (BYOL, SimSiam, DINO);
- Canonical correlation (VICReg, Barlow Twins);
- Maxed image modeling.

The projection introduced in SimCLR is needed because the training task is different from the downstream task.

**On the representation and learning
of monotone triangular transport maps**
R. Baptista et al.

A *Triangular transport map* S between two probability distributions μ and ν on \mathbf{R}^d is a map of the form

$$S : \begin{cases} \mathbf{R}^d \longrightarrow \mathbf{R}^d \\ x \longmapsto \begin{bmatrix} S_1(x_1) \\ S_2(x_1, x_2) \\ \vdots \\ S_d(x_1, \dots, x_d) \end{bmatrix} \end{cases}$$

such that $X \sim \mu \Rightarrow Z = S(X) \sim \nu$, i.e., $\mu = S^\# \nu$ (this is an alternative to normalizing flows).

If the $x_k \mapsto S_k(x_{<k}, x_k)$ are increasing, then S is invertible.

If $\mu \ll \text{Lebesgue}$ and $\nu = N(0, I)$, then S is essentially unique.

The *rectification operator* R_k transforms a smooth function into a monotonic one

$$R_k(f_k)(x_{<k}, x_k) = f_k(x_{<k}, 0) + \int_0^{x_k} g[\partial_k f_k(x_{<k}, t)] dt$$

for some $g : \mathbf{R} \rightarrow \mathbf{R}_{>0}$, e.g., softplus.

To find the S_k 's from data sampled from μ (for the reference distribution $\nu = N(0, I)$), minimize

$$\hat{J}_k(S) = \frac{1}{n} \sum_i \frac{1}{2} S(x_{\leq k}^i)^2 - \log |\partial_k S(x_{\leq k}^i)|$$

(the KL divergence, up to a constant), under the constraint $\partial_k S(x_{\leq k}) > 0$ (use the rectification operator to remove the constraint). Under reasonable assumptions, there are no spurious local minima: they are all global.

Look for S_k of the form

$$\begin{aligned} S_k(x_{\leq k}) &= \sum_{\alpha \in \Lambda} c_\alpha \psi_\alpha(x_{\leq k}) \quad \alpha \text{ multi-index} \\ \psi_\alpha(x_{\leq k}) &= \prod_{j=1}^k \psi_{\alpha_j}(x_j) \end{aligned}$$

where the ψ_α are:

- Hermite polynomials (orthogonal wrt the Gaussian density)

$$\phi_\alpha(x) = (-1)^\alpha e^{x^2/2} \frac{d^\alpha}{dx^\alpha} e^{-x^2/2},$$

modified to be linear outside some compact interval $[a, b]$

- or wavelets

$$\begin{aligned} \psi_{\ell, q}(x) &= 2^{\ell/2} \psi(2^\ell x - q) \\ \psi(x) &= (1 - x^2) e^{-x^2/2} \quad (\text{Mexican hat}). \end{aligned}$$

Greedy build the multi-index set Λ by adding the multi-index α giving the best improvement to \hat{J}_k

$$\Lambda_{t+1} = \Lambda_t \cup \{\alpha_t^*\}$$

ensuring that $(\Lambda_t)_{t \geq 0}$ remains downward closed

$$\forall \alpha \in \Lambda_t \quad \forall \alpha' \leq \alpha \quad \alpha' \in \Lambda_t,$$

i.e., α_t^* is in the reduced margin of Λ_t

$$\alpha_t^* = \underset{\alpha \in \Lambda_t^{\text{RM}}}{\text{Argmin}} \left| \nabla_{\alpha} \hat{J}_k \right|.$$

The sparsity structure of Λ defines a graphical model.

The triangular structure of the transport map

$$S(x_1, x_2) = \begin{bmatrix} S_1(x_1) \\ S_2(x_1, x_2) \end{bmatrix}$$

gives the conditional distribution

$$X_2 | X_1 = x_1 \sim S_2(x_1, \cdot)^{\#} \nu_2.$$

Implementation: MPaT.

Categorical reparametrization with Gumbel softmax E. Jang et al.

Sampling from a categorical distribution

$$z \sim \text{Categorical}(\pi_1, \dots, \pi_k)$$

is equivalent to (*Gumbel max trick*)

$$g_1 \stackrel{\text{iid}}{\sim} \text{Gumbel}(0, 1) \\ z = \underset{i}{\text{one-hot Argmax}}(g_i + \log \pi_i)$$

where the Gumbel distribution can be sampled as

$$u \sim \text{Unif}(0, 1) \\ g = -\log -\log u.$$

The *Gumbel softmax* replaces argmax with softmax.

$$g_1 \stackrel{\text{iid}}{\sim} \text{Gumbel}(0, 1) \\ z = \text{softmax}_{\tau}(g + \log \pi) \in \Delta$$

It is a reparametrization trick for an approximate categorical distribution.

In practice:

- Slowly decrease the temperature τ to approach 1-hot vectors;
- If you need a discrete output (e.g., discrete actions in reinforcement learning), use argmax in the forward pass and the Gumbel softmax in the backward pass.

The concrete distribution: a continuous relaxation of discrete random variables C.J. Maddison et al.

Earlier paper on the Gumbel softmax (initially called “concrete”).

Learning with differentiable perturbed optimizers Q. Berthet et al. (2020)

The optimization problem

$$y^*(\theta) = \underset{y \in \text{Conv } Y}{\text{Argmax}} \langle y, \theta \rangle,$$

for a finite set Y , can be perturbed into

$$y_{\varepsilon}^*(\theta) = \underset{z \sim \mu}{\text{E}} [y^*(\theta + \varepsilon Z)];$$

the Jacobian is

$$J_{\theta} y_{\varepsilon}^*(\theta) = \underset{z \sim \mu}{\text{E}} [y^*(\theta + \varepsilon Z) \nabla_Z \nu(Z)' / \varepsilon]$$

where $d\mu(z) = e^{-\nu(z)} dz$.

This generalizes the Gumbel softmax:

$$\mu = \text{Gumbel}(0, 1) \\ Y = \{\text{1-hot vectors}\} \\ \text{Conv } Y = \Delta$$

Many problems can be put in this form (maximizing a linear quantity over a polytope):

- Ranking: $Y = \{\sigma \cdot v : \sigma \in \mathfrak{S}_n\}$, for some vector v ; $\text{Conv } Y$ is then the *permutohedron*;
- Shortest path, etc.

The perturbed solution y_{ε}^* minimizes the *Fenchel-Young loss*

$$L_{\varepsilon}(\theta; y) = F_{\varepsilon}(\theta) + \varepsilon \Omega(y) - \langle \theta, y \rangle \\ F_{\varepsilon}(\theta) = \underset{Z \sim \mu}{\text{E}} \underset{y \in \text{Conv } Y}{\text{Max}} \langle y, \theta + \varepsilon Z \rangle \\ \Omega = F_1^* \quad (\text{Fenchel dual}).$$

Differentiation of blackbox combinatorial solvers M. Vlastelica et al.

To back-propagate through a combinatorial solver

$$y(w) = \underset{y \in Y}{\text{Argmin}} \langle w, y \rangle$$

(*linear* objective, arbitrary constraints), use the actual (discrete) solution in the forward pass and a continuous interpolation of (a linearization of) the loss in the backward pass.

$$\hat{y} = \text{solver}(\hat{w}) \\ \frac{dL}{dy}(\hat{y}) : \text{gradient to back-propagate}$$

$$w' = \hat{w} + \lambda \frac{dL}{dy}(\hat{y})$$

$$y_{\lambda} = \text{solver}(w')$$

$$\nabla_w f_{\lambda}(\hat{w}) = -\frac{1}{\lambda}(\hat{y} - y_{\lambda})$$

**Optimizing rank-based metrics
with blackbox differentiation**
M. Rolínet et al.

Application of blackbox differentiation to ranking problems, maximizing

- Recall: proportion of relevant items ($y_i^* = 1$) among the top k (for a fixed value of k);
- Average precision: average of recall_k , as k varies;
- Mean average precision: average of the average precision, as the class varies.

$$\begin{aligned} y &\in \mathbf{R}^n && \text{scores} \\ y^* &\in \{0, 1\}^n && \text{ground truth} \end{aligned}$$

**Backpropagation through combinatorial
algorithms: identity with projection works**
S.S. Sahoo et al.

To back-propagate through a blackbox solver $w \mapsto t(w) = \text{Argmin}_{y \in Y} \langle w, y \rangle$, use $-I$, apply a projection P accounting for invariances $\forall w \ y^*(w) = y^*(P(w))$, $d\ell/dy \mapsto -P'(w)d\ell/dy$ and add noise to w before feeding it to the solver.

**Fenchel-Young losses with skewed entropies
for class-posterior probability estimation**
H. Bao and M. Sugiyama (2021)

The optimization problem $\hat{y}(\theta) = \text{Argmax}_{y \in Y} \theta y$ can be regularized into

$$\hat{y}_\Omega(\theta) = \text{Argmax}_y \theta y - \Omega(y).$$

The corresponding *Fenchel-Young loss* is

$$\begin{aligned} \ell_\Omega(\theta, y) &= \Omega^*(\theta) + \Omega(y) - \theta y \\ \Omega^*(\theta) &= \sup_\eta \theta \eta - \Omega(y). \end{aligned}$$

Under reasonable assumptions (Ω strictly convex), ℓ is convex, smooth, and reaches its minimum 0 at $y = \hat{y}_\Omega(\theta)$.

A cdf F defines a Fenchel-Young loss

$$\begin{aligned} \ell_F(\theta, y) &= \Omega_F^* + \Omega_F(y) - y\theta \\ \Omega_F(y) &= \int_0^y F^{-1}(q) dq \\ \Omega_F^*(\theta) &= \int_{-\infty}^\theta F(s) ds. \end{aligned}$$

Because of the symmetry of the logit link, logistic regression tends to underestimate the probability of rare events. Skewed links address that problem but, with maximum likelihood estimation (MLE), they lead to non-convex optimization problems. Instead, use the generalized eigenvalue distribution (GEV) link and the corresponding (convex) Fenchel-Young loss

$$F_\xi(\theta) = \exp -(1 + \xi\theta)_+^{-1/\xi}.$$

Learning with Fenchel-Young losses
M. Blondel et al.

The optimization problem

$$\text{Maximize}_{y \in Y} \langle \theta, y \rangle$$

can be relaxed to

$$\hat{y}_\Omega(\theta) = \text{Argmax}_{y \in \text{Conv } Y} \langle \theta, y \rangle - \Omega(y).$$

Examples include

$$\begin{aligned} \text{Argmax} & \quad \Omega = I_\Delta \\ \text{Softmax} & \quad \Omega = -H + I_\Delta \\ \text{Sparsemax} & \quad \Omega = \frac{1}{2} \|\cdot\|^2 + I_\Delta \\ \text{Sigmoid} & \end{aligned}$$

The *Fenchel-Young loss*

$$\begin{aligned} L_\Omega(\theta, y) &= \Omega^*(\theta) + \Omega(y) - \langle \theta, y \rangle \\ &= f_\theta(y) - f_\theta(\hat{y}_\Omega(\theta)), \end{aligned}$$

where $f_\theta(y) = \Omega(y) - \langle \theta, y \rangle$, is convex, and reaches its minimum at $y = \hat{y}_\Omega(\theta)$.

A *generalized entropy* is a function $H : \Delta_n \rightarrow \mathbf{R}$, strictly concave, \mathfrak{S}_n -symmetric, zero on the corners of Δ_n ; it is maximal for the uniform distribution.

$$\begin{aligned} \text{Shannon} & \quad -\sum p_i \log p_i \\ \text{Tsallis} & \quad \frac{1}{\alpha(\alpha-1)} \left(1 - \sum p_i^\alpha\right) \\ \text{Rényi} & \quad \frac{1}{1-\beta} \log \sum p_i^\beta \\ \text{Norm} & \quad 1 - \|p\|_q \\ \text{Squared norm} & \quad \frac{1}{2} (1 - \|p\|_q^2) \end{aligned}$$

**Differentiable clustering and partial
Fenchel-Young losses**
L. Stewart et al. (2023)

Given a similarity matrix $\Sigma \in \mathbf{R}^{n \times n}$, the adjacency matrix of the k -spanning forest with maximum similarity is the solution of a linear program

$$\text{Maximize}_{A \in C_k} \langle A, \Sigma \rangle$$

where C_k is the set of adjacency matrices of forests with k connected components; it corresponds to single linkage clustering.

**Distributions for compositionally
differentiating parametric discontinuities**
J. Michel et al. (2023)

Integral primitive for differentiable programming languages, to differentiate integrals with discontinuous integrands.

**Cold analysis of
Rao-Blackwellized straight-through
Gumbel-softmax gradient estimator
A. Shekhovtsov (2023)**

There are many estimators of $J = \frac{d}{d\eta} \mathbb{E}_{x \sim p_\eta} [\ell(\phi(x))]$.

$$J^{\text{RF}} = \ell(\phi(x)) \cdot \frac{d}{d\eta} \log p_\eta(x)$$

$$J^{\text{ST}} = \frac{d}{d\eta} \ell(\phi(x)) \cdot \frac{d}{d\eta} \bar{\phi}, \quad \bar{\phi} = \mathbb{E}[\phi(x)]$$

$$J^{\text{DARN}} = \frac{d}{d\eta} \ell(\phi(x)) \cdot (\phi(x) - \bar{\phi}) \cdot \frac{d}{d\eta} \log p_\eta(x)$$

$\bar{\phi}$ = your choice

$$J^{\text{GS}} = \frac{d}{d\phi} \ell(\tilde{\phi}) \cdot \frac{d\tilde{\phi}}{d\eta}$$

$G_k \sim \text{Gumbel}(0, 1)$
 $\tilde{\phi} = \text{softmax}_\ell(\log p_\eta + G)$

$$J^{\text{GS-ST}} = \frac{d}{d\phi} \ell(\phi(x)) \cdot \frac{d\tilde{\phi}}{d\eta}$$

$$J^{\text{GR}} = \frac{d}{d\phi} \ell(\phi(x)) \cdot \mathbb{E}_{G \sim \text{Gumbel}} \left[\frac{d\tilde{\phi}}{d\eta} \right]$$

**Automatic differentiation of programs
with discrete randomness
G. Arya et al.**

Given a family of discrete random variables $X(p) : \Omega \rightarrow E$ for $p \in I = [a, b] \subset \mathbf{R}$, we want another one, $\tilde{X}(p)$, such that

$$\mathbb{E}[\tilde{X}(p)] = \frac{d}{dp} \mathbb{E}[X(p)].$$

We cannot use the reparametrization trick

$$\frac{d\mathbb{E}[X(p)]}{dp} \stackrel{?}{=} \mathbb{E} \left[\frac{dX(p)}{dp} \right]$$

because $dX(p)/dp = 0$ a.s. Instead, let

$$\delta = \lim_{\varepsilon \rightarrow 0} \frac{X(p + \varepsilon) - X(p)}{\varepsilon} = \frac{dX(p)}{dp}$$

$$B > |\delta|$$

$$A_B(\varepsilon) = \left\{ \omega \in \Omega : \left| \frac{X(p + \varepsilon) - X(p)}{\varepsilon} \right| > B \right\}$$

$$= \{\text{large jumps}\}$$

$$w = \frac{dP[A_B(\varepsilon)]}{d\varepsilon} \Big|_{\varepsilon=0} = \frac{dP[\text{large jump}]}{d\varepsilon} \Big|_{\varepsilon=0}$$

$$Y_\varepsilon \stackrel{d}{=} X(p + \varepsilon) - X(p) | A_B(\varepsilon), X(p)$$

$$Y = \lim_{\varepsilon \rightarrow 0} Y_\varepsilon.$$

Then,

$$\frac{d\mathbb{E}[X(p)]}{dp} = \mathbb{E}[\delta + w(Y - X(p))].$$

Concretely, with dual numbers, this gives results like

$$\begin{aligned} X &\sim \text{Bernoulli}(0.6 + \varepsilon) \\ 0 + 1 &\text{ with probability } 2.5\varepsilon \end{aligned}$$

More generally, for random variables with jumps, sampling returns *stochastic triples* (a, b, c, d)

$$a + b\varepsilon + (c \text{ with probability } d\varepsilon).$$

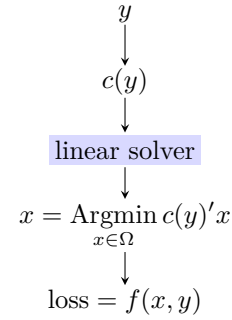
This is amenable to automatic differentiation. Implementation in `StochasticAD.jl`.

**SurCo: learning linear surrogates for
combinatorial nonlinear optimization problems
A. Ferber et al. (2023)**

Solve nonlinear optimization problems

$$\underset{x \in \Omega}{\text{Minimize}} f(x, y)$$

by reducing them to linear problems, by training a neural net to find the closest linear program.



(The linear solver can be an MIP solver.)

**MIPaaL: Mixed integer program as a layer
A. Ferber et al.**

To compute surrogate gradients for an MIP solver, solve the problems by relaxing the integral constraints and progressively adding cutting planes, and use the final (continuous, convex) optimization problem KKT conditions.

**Differentiation of
blackbox combinatorial solvers
M. Vlastelica et al.**

We want to backprop through a blackbox (combinatorial) optimizer with a linear objective.

$$w \mapsto \hat{y} = \underset{y \in Y}{\text{Argmin}} w \cdot \phi(y) \mapsto L(\hat{y})$$

Consider the first order approximation of the loss

$$L(y) \approx L(\hat{y}) + \frac{dL}{dy}(y - \hat{y}) = f(y).$$

Define a relaxation of f

$$y_\lambda(w) = \underset{y \in Y}{\text{Argmin}} w \cdot \phi(y) + \lambda f(y)$$

$$f_\lambda(w) = f(y_\lambda(w)) - \frac{1}{\lambda} [w \cdot \phi(y(w)) - w \cdot \phi(y_\lambda(w))].$$

The surrogate gradient can be computed as

$$w' = \hat{w} + \lambda \cdot \frac{dL}{dy}(\hat{y})$$

$$y_\lambda = \text{solver}(y')$$

$$\nabla_w f_\lambda(\hat{w}) = -\frac{1}{\lambda}(\hat{y} - y_\lambda).$$

Try $\lambda \approx \frac{\langle w \rangle}{\left\langle \frac{dL}{dy} \right\rangle}$ where $\langle \cdot \rangle = \text{avg}$.

Differentiable Monte Carlo samplers with piecewise deterministic Markov processes **R. Seyer**

The ergodic theorem states

$$\mathbb{E}_{X \sim \mu} [f(X)] = \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T f(Z_t) dt$$

but, in general, we do not have

$$\frac{d}{d\theta} \mathbb{E}_{X \sim \mu_\theta} [f(X)] \stackrel{?}{=} \lim_{T \rightarrow \infty} \frac{d}{d\theta} \frac{1}{T} \int_0^T f(Z_t^\theta) dt.$$

To differentiate $\mathbb{E}_{X \sim \mu_\theta} [f(X)]$, there are several approaches:

- The reparametrization trick (aka pathwise derivative) to get a *coupling* between μ_θ and $\mu_{\theta+\varepsilon}$, but it only gives an unbiased estimator if the reparametrization is Lipschitz – this excludes discrete variables or jumps;
- Stratified derivatives, which handle the discontinuities separately (e.g., the “stochastic triples”);
- The score method (REINFORCE).

A *piecewise deterministic Markov process* has 3 components:

- Between jumps, it follows a deterministic flow,

$$\frac{dz}{dt} = \xi(z)$$

for some drift $\xi : E \rightarrow E$;

- The timing of the jumps follow an inhomogeneous Poisson process with rate $\lambda_t = \lambda(z_t)$ for some $\lambda : E \rightarrow \mathbf{R}_+$;
- The jumps are given by a transition kernel $Q(z, dz)$.

The main examples are the *zigzag* sampler on \mathbf{R}

state space	$E = \mathbf{R} \times \{\pm 1\}$
drift	$\xi(x, v) = (v, 0)$
event rate	$\lambda(x, v) = \text{Max}\{0, c\nabla_x - \log \pi(x)\}$
transition kernel	deterministically flip sign(v)

and its generalization, the *bouncy particle* sampler

$$\begin{aligned} E &= \mathbf{R}^d \times \mathbf{R}^d \\ \xi(x, v) &= (v, 0) \\ \lambda(x, v) &= \text{Max}\{0, v' \nabla_x - \log \pi(x)\} \\ Q: &\text{elastic reflection of } v \text{ tangentially} \\ &\text{to the potential } \nabla_x - \log \pi \end{aligned}$$

(there is also a “refreshment rate” in λ , corresponding to resetting the velocity to $v \sim N(0, I)$.)

Those continuous-time non-invertible Markov chains tend to mix faster than the classic discrete-time reversible ones.

To couple the zig-zag processes Z_t^θ and $Z_t^{\theta+\varepsilon}$ on $[0, T_\theta]$ and $[0, T_{\theta+\varepsilon}]$, use the unter-arrival times

$$\tau'_i = \Lambda_t^\leftarrow(\Lambda_i(\tau_i, \theta), \theta + \varepsilon)$$

where

$$\Lambda(t) = \int_0^t \lambda(r) dr$$

$$\Lambda^\leftarrow(\omega) = \inf\{t \geq 0 : \Lambda(t) \geq \omega\} \quad \text{pseudo-inverse}$$

(they have the same number of events). Under reasonable assumptions (Λ invertible, with smooth inverse), we can swap $d/d\theta$ and $\lim_{T \rightarrow \infty}$.

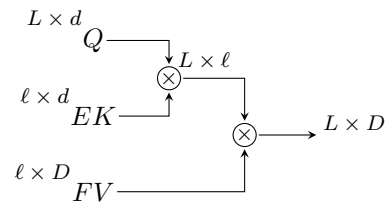
Reformer: the efficient transformer **N. Kitaev et al.**

The attention matrix is quadratic in the context length, but its rows are softmaxes and tend to be sparse: instead of computing the whole matrix, only compute, for each row (query), the k largest entries, corresponding to the k closest keys, with an approximate nearest neighbour (ANN) algorithm, e.g., LSH (locality sensitive hashing).

The Reformer also uses reversible layers.

Linformer: self-attention with linear complexity **S. Wang et al.**

The attention matrix has low rank: reduce the dimension of the key and value matrices before using them.



Random feature attention **H. Peng et al.**

The attention $\text{Att}(q, K, V) = \text{softmax}(qK'/\sqrt{d})V$ is the expectation of a quantity whose computation is linear in time and space; estimating it with a single sample is good enough.

The Gaussian kernel can be estimated as

$$\exp - \frac{\|x - y\|^2}{2\sigma^2} = \mathbb{E}_{w_i \sim N(0, \sigma^2 I)} [\phi(x) \cdot \phi(y)]$$

where

$$\phi : \begin{cases} \mathbf{R}^d & \longrightarrow \mathbf{R}^{2D} \\ x & \longmapsto \begin{bmatrix} \sin(w_1 \cdot x) \\ \vdots \\ \sin(w_D \cdot x) \\ \cos(w_1 \cdot x) \\ \vdots \\ \cos(w_D \cdot x) \end{bmatrix} \end{cases}$$

The softmax appearing in the attention formula can be approximated as

$$\begin{aligned} \exp\left(\frac{x \cdot y}{\sigma^2}\right) &= \exp\left(\frac{\|x\|^2 + ny^2 - nx - y^2}{2\sigma^2}\right) \\ &= \exp\left(\frac{\|x\|^2}{2\sigma^2}\right) \exp\left(\frac{\|y\|^2}{2\sigma^2}\right) \exp\left(\frac{\|x - y\|^2}{2\sigma^2}\right) \\ &= \exp\left(\frac{1}{\sigma^2}\right) \mathbb{E}_w[\phi(x) \cdot \phi(y)] \text{ if } \|x\| = \|y\| = 1 \\ &\approx \exp\left(\frac{1}{\sigma^2}\right) \phi(x) \cdot \phi(y) \end{aligned}$$

The attention operator can therefore be approximated without large intermediate matrices,

$$\begin{aligned} \text{Att}(q, K, V) &= \sum_i \frac{\exp(q \cdot k_i / \sigma^2)}{\sum_j \exp(q \cdot k_j / \sigma^2)} v'_i \\ &\approx \sum_i \frac{\phi(q) \cdot \phi(k_i)}{\sum_j \phi(q) \cdot \phi(k_j)} v'_i \\ &= \sum_i \frac{\phi(q)' \phi(k_i) v'_i}{\sum_j \phi(q)' \phi(k_j)} \\ &= \frac{\phi(q)' \sum_i \phi(k_i) v'_i}{\phi(q)' \sum_i \phi(k_i)} \end{aligned}$$

where σ^2 is the softmax temperature.

Rethinking attention with performers **K. Choromanski et al.**

Attention is defined as

$$\begin{aligned} \text{Att}(Q, K, V) &= D^{-1}AV \\ A &= \kappa(Q, K) = \exp(QK' / \sqrt{d}) \\ D &= \text{diag}(A\mathbf{1}). \end{aligned}$$

The kernel can be written

$$\kappa(x, y) = \mathbb{E}_{\phi}[\phi(x)' \phi(y)]$$

and the attention can be approximated as

$$\begin{aligned} \widehat{\text{Att}}(A, K, V) &= \hat{D}^{-1} \hat{Q} \hat{K}' V \\ \hat{D} &= \text{diag}(\hat{Q} \hat{K}' \mathbf{1}) \\ \hat{Q} &= \phi(Q) \quad (\phi(q'_i)' \text{ as rows}) \\ \hat{K} &= \phi(K) \end{aligned}$$

For the random features ϕ , do not use the trigonometric random features (they have negative values, which increases the variance), but *positive random features*.

$$\phi(x) = \frac{h(x)}{\sqrt{m}} [f_1(\omega'_1 x), \dots, f_1(\omega'_m x), \dots, f_\ell(\omega'_1 x), \dots, f_\ell(\omega'_m x)]$$

$$h(x) = \exp - \frac{\|x\|^2}{2}, \quad \ell = 1, \quad f_1 = \exp$$

$$h(x) = \frac{1}{\sqrt{2}} \exp - \frac{\|x\|^2}{2}, \quad \ell = 2, \quad f_1 = \exp, \quad f_2 = 1/f_1$$

$$\exp(x'y) = \mathbb{E}_{\omega \sim N(0, I)} [\phi(x)' \phi(y)]$$

Use *orthogonal random features* (apply Gram-Schmidt to $\omega_1, \dots, \omega_n$) to further reduce variance.

A survey of transformers **T. Lin et al. (2022)**

Efficient transformers: a survey **Y. Tay et al. (2022)**

List of transformer variants.

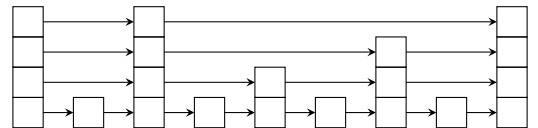
Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs **Y.A. Malkov and D.A. Yashunin**

NSW (navigable small world) adds the points, in a random order, linking each one to its M closest (approximate) neighbours (greedy search with multiple random entry nodes). The construction can be parallelized. The first nodes are better connected than later ones.

HNSW (hierarchical NSW) uses n subgraphs, containing an increasing fraction of the nodes:

- Start the search at the top (smallest) one, at a high degree node;
- Greedily find the node closest to the query;
- Start a new search in the next graph, from this node;
- Continue down the stack of graphs.

This is similar to the *probability skip list* (a stack of sorted linked lists).



Implementation in **faiss**.

Double descent demystified: identifying, interpreting and ablating the sources of a deep learning puzzle **R. Schaeffer et al.**

Double decent for linear regression suggests the phenomenon requires:

- Small but nonzero singular values in the training data features;
- The test data varies in different directions from the training data (it is not in the span of the first singular values);
- The best model possible makes errors on the training data (the training data is noisy).

**Multiple descent:
design your own generalization curve**
L. Chen et al.

**Lost in the middle:
how language models use long contexts**
N.F. Liu et al.

LLMs struggle to use information in the middle of their context: performance is higher when relevant information is at the beginning or end – models with a smaller context perform better.

**Path neural networks:
expressive and accurate graph neural networks**
G. Micher et al. (2023)

Do not aggregate just neighbours, but paths:

- Single shortest paths;
- All shortest paths;
- All single paths of length up to k .

Use recurrent layers to encode paths into vectors.

**Leveraging label non-uniformity for node
classification in graph neural networks**
F. Ji et al. (2023)

Train a GNN for node classification and compute, for each node v , its label non-uniformity

$$w(v) = \left\| \mu_v - \frac{1}{n} \mathbf{1} \right\|_1,$$

where μ_v is the label distribution for v from the model. Nodes with low w are close to class boundaries. To improve model performance,

- Add nodes with low w to the training set, with their *predicted* labels, and retrain;
- Pick a proportion η_1 of nodes with the smallest w ; build a spanning tree on them; drop a proportion η_2 of edges (between those nodes) not in that tree (to preserve connectivity) and re-train.

**Graph positional encoding
via random feature propagation**
M. Eliasof et al. (2023)

Adding node features from the first power iterations combines the advantages of random features (before the first iteration) and spectral positional encoding (limit of the power iterations).

**Towards deep attention in graph neural
networks: problems and remedies**
S.Y. Lee et al. (2023)

GATv2 learns edge attention $\alpha_{ij}^{(k)} \in (0, 1)$, and suffers from over-smoothing; DAGCN learns $\alpha_{ij}^{(k)} \in (-1, 1)$ and does not.

**Understanding convolutions
on graphs via energies**
F. Di Giovanni et al.

The Dirichlet energy is

$$\mathcal{E}(F) = \frac{1}{2} \sum_{ij \in E} \|(\nabla F)_{ij}\|^2$$

$$(\nabla F)_{ij} = \frac{f_j}{\sqrt{d_j}} - \frac{f_i}{\sqrt{d_i}}.$$

It can be generalized to

$$\begin{aligned} \mathcal{E}_\theta(F) &= \sum_i \langle f_i, \Omega f_i \rangle - \sum_{ij} A_{ij} \langle f_i, W f_j \rangle + \phi(F, F(0)) \\ &= \sum_i \langle f_i, (\Omega - W) f_i \rangle + \\ &\quad \frac{1}{2} \sum_{ij} \|\Theta_+(\nabla F)_{ij}\|^2 - \frac{1}{2} \sum_{ij} \|\Theta_-(\nabla F)_{ij}\|^2 + \\ &\quad \phi(F, F(0)) \\ W &= \Theta'_+ \Theta_+ - \Theta'_- \Theta_- . \end{aligned}$$

Graph convolutions do not necessarily oversmooth the data: the negative eigenvalues of W can provide over-sharpening.

**Node embedding from neural Hamiltonian
orbits in graph neural networks**
Q. Kang et al. (2023)

Graphs are not flat: embedding them in hyperbolic space preserves more information. Their *Gromov δ -hyperbolicity distributions* show they do not have constant curvature. Do not embed the nodes in a constant negative curvature space: embed them in (\mathbf{R}^d, g) , where the metric g is learned.

The *Hamiltonian GNN*

- Learns an embedding $(q_n^{(0)}, p_n^{(0)}) \in \mathbf{R}^{2d}$ for each node n ;
- Learns a metric g_{ij} on \mathbf{R}^d (with inverse g^{ij});
- Moves those embeddings along the Hamiltonian flow

$$\begin{aligned} H(p, q) &= \frac{1}{2} g^{ij}(q) p_i p_j \\ \dot{q}_i &= g^{ij} p_j \\ \dot{p}_i &= -\frac{1}{2} \partial_i g^{jk} p_j p_k \\ (q_n^{(0)}(0), p_n^{(0)}(0)) &\rightsquigarrow (q_n^{(0)}(T), p_n^{(0)}(T)) \end{aligned}$$

- Aggregates the resulting embeddings

$$q_n^{(1)} = q_n^{(0)}(T) + \text{Mean}_{m \in \mathcal{N}(n)} q_m^{(0)}(T)$$

(the momentum of the next layer, $p_n^{(1)}$ is learned – it does not come from $p_n^{(0)}(T)$);

The Gromov δ -hyperbolicity of a graph G is

$$\max_{\substack{a,b,c,d \in V \\ d(a,b)+d(c,b) \geq \\ d(a,c)+d(b,d) \geq \\ d(a,d)+d(b,c)}} [d(a,b) + d(d,c)] - [d(a,c) + d(b,d)]$$

Implementation (slightly more efficient than the naive $O(n^4)$) in `sage.graphs.hyperbolicity.*`.

***Can large language models
reason about program invariants***
K. Pei et al. (2023)

Alternative to dynamic analysis tools (e.g., Daikon, for Java, which needs access to several program traces).

***Data representations' study
of latent image manifolds***
I. Kaufman and O. Azencot (2023)

To assess the geometry of latent representations learned by a neural net:

- Compute its intrinsic dimension;
- Augment the data around one of the training samples, to have a dense neighbourhood: for image data, consider the image as a width×height matrix (separately for each channel), compute its SVD, and remove some of the 10 smallest singular values – this produces 2^{10} points;
- Compute the mean absolute principal curvature (MAPC).

The curvature increases sharply in the first few layers, then plateaus, and (only in well-trained networks) increases again in the last layers.

***Intrinsic dimension of data representations
in deep neural networks***
A. Ansuini et al.

To measure the *intrinsic dimension* (ID) of a cloud of points, fit a Pareto distribution to the ratios $\mu_i = r_i^{(2)}/r_i^{(1)}$, where $r_i^{(1)}$ and $r_i^{(2)}$ are the distances between a point i and its first and second nearest neighbours.

$$p(\mu_i) = d \cdot \mu_i^{-(d+1)}$$

$$p(\mu) = d^N \prod_{1 \leq i \leq N} \mu_i^{-(d+1)}$$

It is robust, even if the points are not really sampled uniformly, provided $d \leq 20$ – otherwise it underestimates the dimension. Measure it at different scales (by decimating the data) to check if it is scale-invariant.

The ID of the latent representations in trained neural nets are much lower (10 to 100 times) than the number of neurons; it increases sharply in the first layers, then decreases slowly (“hunchback” shape). The ID of the last layer is a good predictor of performance (lower is better).

The data manifolds are not flat: they cannot be recovered by linear projections.

***Estimating the intrinsic dimension of datasets
by a minimal neighbourhood information***
E. Facco et al.

Since $F_{\text{Pareto}}(\mu) = (1 - \mu^{-d})\mathbf{1}_{[\mu \geq 1]}$, the intrinsic dimension can also be estimated with an (intercept-free) linear regression

$$\frac{\log[1 - F(u)]}{\log \mu} = d.$$

Curvature-aware manifold learning
Y. Li (2017)

To estimate the curvature of a set of points:

- Compute its intrinsic dimension;
- Take a neighbourhood of the point of interest (100 times as many points as the dimension);
- Use PCA to find local coordinates;
- Compute a quadratic approximation of the chart of the manifold in those coordinates;
- The eigenvalues of the Hessian of that quadratic approximation are the *principal curvatures*;
- The mean absolute principal curvature (MAPC) is the average of their absolute values (this is an extrinsic notion of curvature, for submanifolds of a Riemannian manifold).

LLE (local linear embedding) can be generalized to account for (intrinsic) curvature.

Dink-Net: Neural clustering on large graphs
Y. Liu et al. (2023)

Learn node embeddings with a contrastive loss, and cluster the nodes, with a “shrinking” loss bringing the node representations close to the cluster centers, and a “dilation” loss keeping the centers apart, end-to-end.

***Dirichlet diffusion score model
for biological sequence generation***
P. Avdeyev (2023)

Diffusion process on the probability simplex to generate discrete sequences (sudoku puzzles, sudoku solutions, DNA sequences). Jacobi diffusion

$$dx = \frac{s}{2} [a(1-x) - bx] dt + \sqrt{sx(1-x)} dW$$

has $\text{Beta}(a, b)$ as stationary distribution. This can be generalized to a diffusion with a Dirichlet stationary distribution (stick-breaking construction with $k-1$ Jacobi diffusions).

Conformal inference is (almost) free for neural networks trained with early stopping
Z. Liang et al. (2023)

Conformal inference for models selected by early stopping requires a 3-way split of the data:

- To train the models;
- To decide when to stop;
- For conformal inference.

It is possible to use only two subsets, by storing all (or 100) the models, delaying the model selection, and choosing one, anew, for each new observation.

Why random pruning is all we need to start sparse
A. Gadhikar et al. (2023)

Randomly pruned (Erdős-Rényi) neural networks contain a lottery ticket.

Synthetic data for model selection
A. Shishan et al.

Synthetic data is often used for data augmentation (StyleGAN, GAN-GP). You can also use it instead of the validation set, for model selection (early stopping, random seed choice, hyperparameter optimization). Reweight the synthetic dataset using a holdout set of classifiers.

Large language models as optimizers
C. Yang et al.

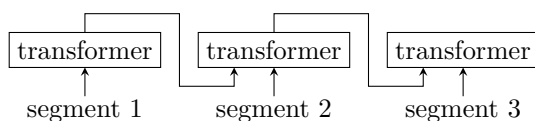
Ask an LLM to generate a prompt template for the problem you want to solve – not just once, but iteratively, each time providing the previous prompts and their performance, to progressively improve it.

Ask for several prompts, each time, for more exploration. Change the LLM sampling temperature to tune the exploration-exploitation tradeoff.

Applications include linear regression, traveling salesman problem, reasoning (GSM8k, BigBenchHard: arithmetic, commonsense).

Scaling transformer to 1M tokens and beyond with RMT
A. Bulatov et al.

To increase the context length of a transformer, split the input into segments, and concatenate the latent representation of segment $i - 1$ to segment i .



Recurrent memory transformer
A. Bulatov et al.

RWKV: reinventing RNNs for the transformer era
A. Albalak et al.

Time mixing

$$\begin{aligned} r_t &= W[\mu x_t + (1 - \mu)x_{t-1}] \\ k_t &= W[\mu x_t + (1 - \mu)x_{t-1}] \\ v_t &= W[\mu x_t + (1 - \mu)x_{t-1}] \\ w_t &= \frac{\sum_{i < t} e^{-(t-1-i)w+k_i} v_i + e^{u+k_t} v_t}{\sum_{i < t} e^{-(t-1-i)w+k_i} + e^{u+k_t}} \\ o_t &= W[\sigma(r_t) \odot w_t] \end{aligned}$$

followed by channel mixing

$$\begin{aligned} r_t &= W[\mu x_t + (1 - \mu)x_{t-1}] \\ k_t &= W[\mu x_t + (1 - \mu)x_{t-1}] \\ o_t &= \sigma(r_t) \odot [W \text{Max}(k_t, 0)^2]. \end{aligned}$$

Precise zero-shot dense retrieval without relevance labels
L. Gao et al.

Instead of computing the similarity between a query and the documents in the database, HyDE (hypothetical document embeddings) asks an LLM to produce a hypothetical (potentially hallucinated) document answering the query, and retrieve the (actual) documents most similar to the hypothetical document.

Pysentimiento: a Python toolkit for sentiment analysis and social NLP tasks
J.M. Pérez et al.

Transformer-based models (RoBERTa) for sentiment and emotion detection, for en, es, it, pt.

Visualizing graph neural networks with CorGIE: corresponding a graph to its embedding
Z. Liu et al.

To assess the quality of the node embeddings learned by a GNN, compare the distance matrices of:

- The latent representations (node embeddings), using the cosine distance;
- The graph, with the inverse Jaccard index of the k -hop neighbours;
- The node features, with the Euclidean distance.

Supervised probabilistic principal component analysis

Probabilistic PCA (PPCA) estimates the parameters of the data generation process

$$\begin{aligned} z &\sim N(0, I) && \text{latent} \\ \varepsilon &\sim N(0, \sigma^2 I) \\ x &= Wz + \varepsilon && \text{observed} \end{aligned}$$

Supervised probabilistic PCA (SPPCA) adds another observed variable.

$$\begin{aligned} z &\sim N(0, I) && \text{latent} \\ \varepsilon &\sim N(0, \sigma_1^2 I) \\ \eta &\sim N(0, \sigma_2^2 I) \\ x = W_1 z + \varepsilon && \text{observed} && y = W_2 z + \eta && \text{observed} \end{aligned}$$

It can be estimated with EM.

A new approach to decomposition of economic time series into permanent and transitory components with particular attention to measurement of the business cycle
S. Beveridge and C.R. Nelson

SAITS: self-attention-based imputation for time series
W. Du et al. (2023)

Impute missing values in time series with diagonally masked self-attention, and two losses, for:

- Reconstruction of artificially masked values;
- Reconstruction of observed values (which may be different from the original ones, as the network also tries to “denoise” the data).

Also check: GRUI-GAN, E²GAN, M-RNN, BRITS, GP-VAE.

BRITS: bidirectional recurrent imputation for time series
W. Cao et al.

Bidirectional RNN for the imputation of missing values in time series, with a consistency loss (the reconstructed values in the forward and backward directions should be close).

Learning representations for incomplete time series clustering
Q. Ma et al. (2021)

CRLI combines:

- (Bidirectional) RNN to impute missing values and compute a latent representation of the time series (concatenation of the last hidden states of the forward and backward RNNs);
- Discriminator, predicting which values were imputed;
- Decoder of the latent representation, attempting to reconstruct the original time series;
- Soft- k -means loss on the latent representation H :

$$\ell(H) = \min_{F'F=I} \text{tr}(H'H) - \text{tr}(F'H'HF)$$

(F is the truncated SVD of H).

Graph-guided network for irregularly sampled multivariate time series
X. Zhang et al.

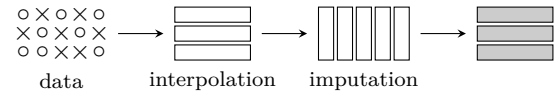
To classify irregularly sampled multivariate time series (or perform some other downstream task), compute latent representations of each time series at each time step:

- With a (feature-specific) nonlinear transformation, if it is observed;
- As a combination of its neighbours’ features (cross-attention) otherwise.

The graph structure used to define the neighbours is initialized as a complete graph; the edge weights are updated using the cumulated attention weight for each edge; the least important edges (bottom $k\%$) are pruned.

Estimating missing data in temporal streams using multi-directional recurrent neural networks
J. Yoon et al. (2017)

M-RNN first interpolates the missing values with a bidirectional RNN, separately for each time series, then refines the imputed values with a cross-sectional fully-connected network.



Powerful p -value combination methods to detect incomplete association
S. Yoon et al. (2021)

To combine p -values,

- The *Fisher* method converts them to χ^2 :

$$\begin{aligned} p_i &\stackrel{\text{iid}}{\sim} U(0, 1) \\ T &= -2 \sum \log p_i \sim \chi^2(2n); \end{aligned}$$

- The *Lancaster* method uses the sample sizes s_i as weights

$$L = \sum F_{\chi^2(s_i)}^{-1}(p_i) \sim \chi^2(\sum s_i)$$

but the excessive degrees of freedom decrease the power;

- The *weighted Fisher* method rescales the sample sizes to keep the same degrees of freedom, using $\chi^2(\text{df} = 2n) = \Gamma(k = n, \theta = 2)$;
- The *roP method* uses the r -th ordered (smallest) p -value

$$p_{(r)} \sim \text{Beta}(r, n - r + 1)$$

but it is very sensitive to the choice of r ;

- *OrdMeta* uses $x = \min_r F_{(r)}(p_{(r)})$ as statistic

$$p = 1 - n! \int_{F_{(n)}^{-1}}^1 \cdots \int_{F_{(2)}^{-1}}^{t_3} \int_{F_{(1)}^{-1}}^{t_2} dt_1 \dots dt_n.$$

R implementation in `metapro`, with `SymPy` for the integral.

Combining dependent p-values with an empirical adaptation of Brown's method
W. Poole et al. (2016)

To combine p -values, the Fisher method uses

$$p_i \stackrel{\text{iid}}{\sim} U(0, 1)$$

$$\Psi = \sum -2 \log p_i \sim \chi^2(2k).$$

If the p -values are not independent, the Brown (Kost) method posits $\Psi \sim c \cdot \chi^2(2f)$, estimates c and f by matching the first two moments

$$f = \frac{E[\Psi]^2}{\text{Var}[\Psi]}$$

$$c = \frac{\text{Var}[\Psi]}{2E[\Psi]}$$

$$E[\Psi] = 2k$$

$$\text{Var}[\Psi] = 4k + 2 \sum_{i < j} \text{Cov}[-2 \log p_i, -2 \log p_j]$$

and approximates the covariance as

$$\text{Cov}[-2 \log p_i, -2 \log p_j] = a\rho_{ij} + b\rho_{ij}^2 + c\rho_{ij}^3$$

$$a = 3.263 \quad b = -.719 \quad c = 0.027$$

$$\rho_{ij} = \text{Cor}(X_i, X_j)$$

$$X_i = \Phi^{-1}(p_i)$$

(in particular, this also assumes a Gaussian copula).

Instead, approximate the covariance with the ecdf.

How good are convex hull algorithms?
D. Avis et al. (1996)

There are several convex hull algorithm, each library implementing only one of them:

- Double description: `cddlib`, `porta`,
- Reverse search: `lrslib`,
- Quick hull: `qhull`.

Each of them has trouble dealing with some type of polytopes:

- Fat lattice: $\#\text{facets} \gg \text{size}$;
- Intricate: $\#\text{triangles} \gg \text{size}$;
- Dwarfed: $P = Q \cap H$, $\#\text{facets}_P < \#\text{facets}_Q$, $\text{size}_Q \gg \text{size}_P$, where the size is the space required to store both representations of the polytope (set of vertices, set of half-spaces) and \gg means “super-polynomial”.

Quasi-stable coloring for graph compression
M. Kayali and D. Suciu

The Weisfeler-Leman (WL) algorithm (aka colour refinement) produces a *stable colouring*, which can be used for graph reduction (merge all the nodes with the same colour), but the reduction is rather modest. Quasi-stable colourings allow two nodes to have the same colour if the number of neighbours they have in each colour is at most q . Applying the WL algorithm until there are n colours is a good heuristic.

Applications include betweenness centrality, maximum flow.

Adaptive hierarchical regular binning of data point features
D. Floros et al.

Efficient construction of a k -d-tree from the dyadic encoding of the point coordinates.

QUBO.jl: a Julia ecosystem for quadratic unconstrained binary optimization
P.M. Xavier et al.

Many optimization problems can be converted (automatically) to QUBO form (quadratic unconstrained binary optimization), which can be solved by many quantum computers.

Learning QUBO models for quantum annealing: a constraint-based approach
F. Richoux et al.

To convert a combinatorial optimization problem to QUBO form, we need to convert the constraints into quadratic terms in the objective function.

- For the 1-hot constraint, since $\sum x_i = 1 \Leftrightarrow (\sum x_i - 1)^2 = 0$, we can add $(\sum x_i - 1)^2$;
- Permutation constraints are handled similarly.

$$\forall \sigma \in \{0, 1\}^{n \times n} \quad \sigma \in \mathfrak{S}_n \Leftrightarrow \sigma \mathbf{1} = \mathbf{1} \wedge \sigma' \mathbf{1} = \mathbf{1}$$

Multi-fidelity covariance estimation in the log-Euclidean geometry
A. Maurais et al. (2023)

The Euclidean multi-fidelity covariance estimator

$$\hat{\Sigma}_n^{(0)} + \sum_{\ell} \alpha_{\ell} (\hat{\Sigma}_{n_{\ell}}^{(\ell)} - \hat{\Sigma}_{n_{\ell-1}}^{(\ell)})$$

is not guaranteed to be positive definite. Replace the Euclidean geometry with the log-Euclidean one (*i.e.*, use $\text{Log } \Sigma$ instead of Σ , and Fréchet average); the Bures-Wasserstein geometry, or the log-Cholesky geometry are alternative choices. (One can compute the optimal sizes n^* and the optimal weights α^* to maximize the MSE with a constraint on the total cost.)

The generalized variance and correlation are

$$\sigma_X^2 = \text{tr Cov}[XX']$$

$$\rho_{XY} = \frac{\text{tr Cov}[XX', YY']}{\sigma_X \sigma_Y}.$$

**Correlation matrix clustering
for statistical arbitrage portfolios
A. Cartea et al. (2023)**

- Take daily returns, for 600 stocks, over 4 years;
- Remove the market (adjusting for the CAPM beta);
- Compute the correlation matrix C ;
- View it as a weighted graph, with potentially negative weights; compute the Laplacians

$$\bar{A}_{ij} = |A_{ij}|$$

$$\bar{D}_{ii} = \sum_j |A_{ij}|$$

$$D_{ii} = \sum_j A_{ij}$$

$$L = \bar{D} - \bar{A}$$

$$\bar{L}_{rw} = I - \bar{D}^{-1}A$$

$$\bar{L}_{sym} = I - \bar{D}^{-1/2}A\bar{D}^{-1/2}$$

$$A = A^+ - A^-$$

$$D_{ii}^+ = \sum_j A_{ij}^+, \quad D_{ii}^- = \sum_j \bar{A}_{ij}$$

$$L^\pm = D^\pm - A^\pm$$

$$L_{sym}^\pm = (D^\pm)^{-1/2}L^\pm(D^\pm)^{-1/2}$$

$$(L_1, L_2) = (L^+ + \tau D^-, L^- + \tau D^+), \quad \tau > 0$$

$$(L_{1,sym}, L_{2,sym}) = (L_{sym}^+ + \tau I, L_{sym}^- + \tau I)$$

- Reduce the dimension using the k smallest (generalized) eigenvalues of \bar{L} or \bar{L}_{rw} or \bar{L}_{sym} or (L_1, L_2) or $(L_{1,sym}, L_{2,sym})$;
- Cluster the resulting points using k -means (with the same k);
- Choose k using random matrix theory (RMT), as the number of eigenvalues of C beyond $\lambda_+ = (1 + \sqrt{N/T})^2$, or the proportion of variance explained by the top eigenvalues of C .

**Is there a replication crisis in finance?
T. Engerslev et al. (2022)**

Apparently, no:

- Do not use returns, but risk-adjusted returns (CAPM alpha);
- Adjust for multiple testing, either with BY or, better, a Bayesian model, to account for correlation between factors (set the prior expected alpha to zero, and select the other with empirical Bayes, using only out-of-sample data to overcome the publication bias);
- Use that Bayesian model to compute the FDR.

**Constrained monotonic neural networks
D. Runje and S.M. Shankaranarayana (2023)**

Neural networks with positive weights and 3 activation functions (in different channels) $\not\prec \not\bowtie \not\bowtie$ can approximate any monotonic function (with ReLU alone,

we only get convex functions, and with $\not\prec$ alone, the network is difficult to train).

Previous solutions used complicated architectures (lattice networks), or a gradient penalty followed a search for non-monotonicity (MILP, SMT).

**How to address monotonicity
for model risk management
D. Chen and W. Ye**

There are several notions of monotonicity between features (e.g., past and present values of the same feature)

$$\forall c \geq 0 \quad f(x) \leq f(x + c)$$

$$\forall c \geq 0 \quad f(x, x + c) \leq f(x + c, x)$$

$$\forall c \geq 0 \quad f(x, y + c) \leq f(x + c, y)$$

(for instance, the probability of recidivism is strongly monotonic wrt felonies and dismeanors). For individual or weak monotonicity, used a GAM with $f'_i \geq 0$ or $f'_i \geq f'_j$. For strong pairwise monotonicity, use a GAM or a NAM (neural additive model), with univariate and bivariate features (groves of NAMs, GNAM) and penalties to ensure the required monotonicities.

**LinSATNet:
the positive linear satisfiability neural networks
R. Wang et al. (2023)**

The *Sinkhorn algorithm* converts a positive matrix $S \in \mathbf{R}_{\geq 0}^{m \times n}$ to a doubly stochastic one by alternatively normalizing its rows and columns to match marginal distributions $u \in \mathbf{R}_{\geq 0}^m, v \in \mathbf{R}_{\geq 0}^n$. It can be generalized to multiple sets of marginal distributions (it still converges).

To enforce constraints $Ax \leq B, Cx \geq D, Ex = F$, where A, B, C, D, E, F have nonnegative entries, to some input y , apply the generalized Sinkhorn algorithm to S

$$S = \exp(W/\tau)$$

$$W = \begin{bmatrix} y' & \beta \\ \beta \mathbf{1}' & \beta \end{bmatrix} = \left[\begin{array}{c|c} y_1 \cdots y_\ell & \beta \\ \hline \beta & \beta \end{array} \right]$$

with marginals encoding the constraints

$$a'x \leq b \quad u = [a' \ b] \quad v = \begin{bmatrix} b \\ a' \mathbf{1} \end{bmatrix}$$

$$c'x \geq d \quad u = [c' \ \gamma d] \quad v = \begin{bmatrix} (\gamma + 1)d \\ c' \mathbf{1} - d \end{bmatrix} \quad \gamma = \left\lfloor \frac{c' \mathbf{1}}{d} \right\rfloor$$

$$e'x = f \quad u = [e' \ 0] \quad v = \begin{bmatrix} f \\ e' \mathbf{1} - f \end{bmatrix}$$

**Autocoreset: an automatic practical
coreset construction framework
A. Maalouf et al.**

A *coreset* is a (weighted) subset of the training data on which the loss function takes similar values. Given a matrix $M \in \mathbf{R}^{n \times m}$, a vector summarization ε -coreset

is a subset $J \subset [n]$ and a weight function $w : J \rightarrow \mathbf{R}_+$ such that

$$\left\| \sum_{i \in [n]} M_i - \sum_{j \in J} w(j) M_j \right\|_2^2 \leq \varepsilon.$$

Given

$$\begin{aligned} &\theta_1, \dots, \theta_m \\ \text{Data} &= \{x_1, \dots, x_n\}, \\ M_{ij} &= \ell(x_i, \theta_j) \end{aligned}$$

iterate

$$\begin{aligned} (I, w) &= \text{Coreset}_m(M) \\ \theta^* &= \underset{\theta}{\text{Argmin}} \sum_{i \in I} w(i) f(x_i, \theta) \\ M_{i, m+1} &= \ell(x_i, \theta^*). \end{aligned}$$

***Lowering the pre-training tax
for gradient-based subset training:
a lightweight distributed pre-training toolkit***
Y. Ro et al. (2023)

Pretrain using all the data, in a distributed fashion, as a model soup, with each model trained independently on a subset of the data (to limit communication); then, find a subset of the data with the same gradient distribution as the whole data, and finish training on that.

***Compositional exemplars
for in-context learning***
Y. Ye et al. (2023)

Determinantal point process (DPP) to select (with maximum a posteriori (MAP) estimators) examples for few-shot learning.

$$\begin{aligned} P(S) &= \frac{\det L_S}{\det(L_I)} \\ L_{ij} &= k(a_i, a_j) \end{aligned}$$

modified to include a relevance term:

$$\begin{aligned} a_i &: \text{exemplar} \\ x &: \text{task} \\ r_i &= g(a_i, x) \text{ relevance of } a_i \text{ to } x \\ \tilde{L} &= \text{diag}(r) L \text{diag}(r) \\ \log \det \tilde{L}_S &= \sum_{i \in S} \log r_i^2 + \log \det L_S \end{aligned}$$

DPPy: DPP sampling with Python
G. Gautier et al. (2019)

***Data-driven subgroup identification
for linear regression***
Z. Izzo et al. (2023)

To find a region where the relation $x \mapsto y$ is almost linear,

- Fit a linear model on the k nearest neighbours of each point;
- Pick the model with the lowest training error
- Enlarge the neighbourhood to include all points with a low error for that model
- Pick the largest ball (or polytope) containing only points in that subset.

***XTab: cross-table pretraining
for tabular transformers***
B. Zhu et al. (2023)

Self-supervised learning for tabular data can use the following losses:

- Reconstruction: reconstruct a corrupted row (60% of the entries have been replaced by resampling from the corresponding column)
- Contrastive loss: the positive pairs are (x, \tilde{x}) , where x is a row and \tilde{x} a corrupted version of it; the negative pairs are different rows.
- Supervised loss: predict the value of a column.

You can pretrain a single model and fine-tune it on different tables and tasks:

- Featurize a row into a sequence of “tokens” (embeddings), ending with CLS: for qualitative columns, learn an embedding, for quantitative columns, learn an affine transformation $\mathbf{R} \rightarrow \mathbf{R}^d$;
- Feed it to an FT-transformer (or some other transformer variant);
- Finish with a projection head.

The transformer is shared across tasks and frozen after pretraining; the featurizers and projection heads are task-specific and have to be re-trained.

***TransTab: learning transferable tabular
transformers across tables***
Z. Wang and J. Sun

Earlier paper with a similar idea and a *gated transformer*, *i.e.*, a transformer followed by a gated feedforward layer: $x \mapsto W_1(W_2 \odot x \oplus W_3x)$.

***Revisiting deep learning models
for tabular data***
Y. Gorishniy et al.

The FT-Transformer tokenizes the rows (*i.e.*, converts them to sequences of embeddings) before feeding them to the transformer:

- For qualitative columns, learn an embedding of the possible values;
- For quantitative columns, learn an affine transform $\mathbf{R} \rightarrow \mathbf{R}^d$;
- Append a CLS token,

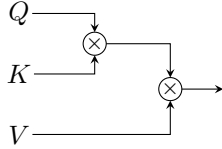
Also try a ResNet.

Fastformer:
additive attention can be all you need
 C. Wu et al.

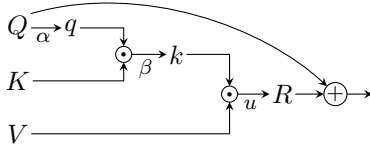
Efficient transformer variants use:

- Hashing (Reformer)
- Low-rank factorizations (Linformer)
- A mixture of local and global attention (Longformer, BigBird)

The FastFormer reduces the dimension of the attention matrix.



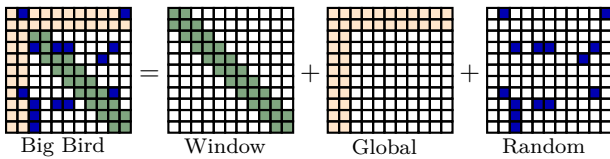
by adding linear transformations.



Big Bird: *transformers for longer sequences*
 M. Zaheer et al.

BigBird makes transformers more scalable by combining:

- Moving window attention;
- Sparse (random) attention – random graphs are good approximations of complete graphs;
- Global attention, either allowing a few random tokens to attend to all other tokens, or by adding a few CLS tokens.



BigBird is still a Turing-complete universal approximator.

SAINT: *improved neural networks for tabular data via row attention and contrastive pre-training*
 G. Somepalli et al.

Transformer on tabular data, alternating between:

- Within-row attention,
- Between-row attention, concatenating the embeddings of all the columns (limited to rows in the current batch).

Pre-train in a self-supervised way (contrastive learning with the InfoNCE loss and/or denoising).

CoDi: *co-evolving contrastive diffusion models for mixed type tabular synthesis*
 C. Lee et al. (2023)

The *diffusion probabilistic model* can be defined for continuous

$$q(x_t|x_{t-1}) = N(x_t; \sqrt{1 - \beta_t}x_{t-1}, \beta_t I)$$

$$p_\theta(x_{t-1}|x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t))$$

or discrete variables

$$q(x_t|x_{t-1}) = (1 - \beta_t)\mathbf{1}_{x_t=x_{t-1}} + \beta_t \text{Unif}(x_t)$$

$$p_\theta(x_{t-1}|x_t) = \sum_{x_0} q(x_{t-1}|x_t, x_0) p_\theta(x_0, x_t)$$

$$q(x_{t-1}|x_t, x_0) = (\text{Bayes}).$$

Both minimize

$$\text{KL}(q(x_{t-1}|x_t, x_0) \| p_\theta(x_{t-1}|x_t)).$$

For tabular data, use two models, one continuous, one discrete, each conditioned on the other one at each step.

TabDDPM: *modeling tabular data with diffusion models*
 A. Kotelnikov

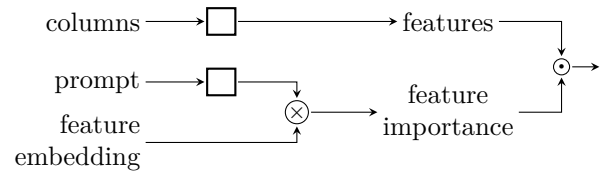
Similar idea, with a quantile transformation of quantitative columns.

Compressing tabular data via latent variable estimation
 A. Montanari and E. Weiner (2023)

To compress tabular data, first cluster the rows and columns (*k*-means on the SVD).

Tromp: *towards a better deep neural network for tabular data*
 K.Y. Chen et al. (2023)

Use task- (prompt-) specific feature importances.



Learning deep time-index models for time series forecasting
 G. Woo et al. (2023)

Learn a “time-index” model, *i.e.*, a 1-dimensional implicit neural representation (INR, NeRF – these generalize the classical structural time series models decomposing a time series into trend, cycle and holiday components)

$$\hat{y}_t = f_{\theta, \phi}(\tau(t))$$

where τ are random Fourier features and the parameters are separated into curve-fitting θ and forecasting/extrapolating (inductive bias) ϕ :

$$\begin{aligned}\theta &\leftarrow \underset{\theta}{\text{Argmin}} \text{fitting error}(\theta, \phi) && \text{on } [t-h, t] \\ \phi &\leftarrow \underset{\phi}{\text{Argmin}} \text{forecasting error}(\theta, \phi) && \text{on } [t, t+h].\end{aligned}$$

Non-autoregressive conditional diffusion models for time series prediction
L. Shen and J.T. Kwok (2023)

TimeDiff forecasts a time series using a diffusion model using, as conditioners:

- An AR model of the past, computed as

$$\sum w_i \odot x_{t-i} + b;$$

- A mixture of the past, $x_{\leq t}$, transformed with a conv-net, and the future, $x_{>t}$ (a kind of teacher forcing).

The sigmoid-weighted linear unit is

$$\text{SiLU}(z) = \sigma(z) \odot z.$$

Also check: CSDI, SSSD, FedFormer, NBeats.

CSDI: conditional score-based diffusion models for probabilistic time series imputation
Y. Tashiro et al. (2021)

Diffusion model to impute the missing values in a time series, conditioning on the observed values, trained with self-supervised learning.

Efficiently modeling long sequences with structured state spaces
A. Gu et al.

The three ways of representing a state space model (SSM), continuous time, recurrent, and convolutional, are related.

$$\begin{aligned}x' &= Ax + Bu \\ y &= Cx \\ x_k &= \bar{A}x_{k-1} + \bar{B}u_k \\ y_k &= Cx_k \\ \bar{A} &= (I - \tfrac{1}{2}hA)^{-1}(I + \tfrac{1}{2}hA) \\ \bar{B} &= (I - \tfrac{1}{2}hA)^{-1}hB \\ y &= K * u \\ K &= (C\bar{A}^0\bar{B}, C\bar{A}^1\bar{B}, \dots, C\bar{A}^{n-1}\bar{B}, C\bar{A}^n\bar{B})\end{aligned}$$

With the HiPPO matrix

$$A_{nk} = - \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n & \text{if } n = k \\ 0 & \text{otherwise} \end{cases}$$

the SSM (S4) memorizes its input u .

It is possible to efficiently compute the convolution with K if A is normal ($AA^* = A^*A$; equivalently, $A = UDU^*$, D diagonal, $UU^* = I$). The HiPPO matrix is the sum of a normal and a low-rank matrix; it is still possible to efficiently compute the convolution with K . With a change of basis, we can replace the normal matrix with a diagonal one.

HiPPO: recurrent memory with optimal polynomial projections
A. Gu et al. (2020)

To compress the information contained in a function $f|_{[0,t]}$, online (*i.e.*, progressively, updating the compressed information as t increases), project it on a finite-dimensional subspace, e.g., that of polynomials of degree less than N , for some measure μ_t on $[0, t]$,

$$g_t = \underset{\substack{g \in \mathbf{R}[X] \\ \deg g < N}}{\text{Argmin}} \left\| f_{\leq t} - g \right\|_{L^2(\mu_t)}.$$

The measure μ_t defines orthogonal polynomials $(g_n)_{n < N}$; the coefficients of g_t in this basis are

$$c_k(t) = \langle f_{\leq t}, g_k \rangle_{\mu_t}.$$

They are solution of an ODE.

$$\frac{dc(t)}{dt} = A_t c(t) + B_t f(t)$$

For instance:

- Legendre (LMU, Legendre memory unit)

$$\begin{aligned}\mu_t &= \frac{1}{\theta} \mathbf{1}_{t-\theta, t} \\ A_{nk} &= \frac{1}{\theta} \begin{cases} (-1)^{n-k}(2n+1) & \text{if } n \geq k \\ (2n+1) & \text{if } n \leq k \end{cases} \\ B_n &= \frac{1}{\theta} (2n+1)(-1)^n\end{aligned}$$

- Laguerre

$$\begin{aligned}\mu_t(x) &= e^{-(t-x)} \mathbf{1}_{x \leq t} \\ A_{nk} &= \mathbf{1}_{n \geq k} \\ B_n &= 1\end{aligned}$$

- Scaled Legendre

$$\begin{aligned}\mu_t &= \frac{1}{t} \mathbf{1}_{[0,t]} \\ A_{nk} &= \begin{cases} (2n+1)^{1/2}(2k+1)^{1/2} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases} \\ B_n &= (2n+1)^{1/2}\end{aligned}$$

- Fourier recurrent unit (FRU): orthogonal polynomials on the unit circle $\{z : |z| = 1\}$.

This can be used inside an RNN, to increase memory.

It's raw!

Audio generation with state space models
K. Goel et al.

UNet with S4 blocks for audio generation; can be bidirectional.

S4 is a state-space model, not computed iteratively, but all at once, with a convolution – this can be done efficiently if the matrix is diagonal-plus-low-rank (1 or 3); it is initialized with a HiPPO matrix. Parametrizing the matrix as $\Lambda - pp^*$, Λ diagonal, $p \in \mathbf{R}^{N \times r}$, instead of $\Lambda + pq^*$, ensures it is Hurwitz (its eigenvalues have a negative part), *i.e.*, the corresponding SSM is stable.

Diffusion-based time series imputation and forecasting with structured state space models
J.M. Lopez and N. Strodthoff

SSSD is a diffusion model with S4 blocks for missing value imputation in time series.

Deep latent state space models for time series generation
L. Zhou et al.

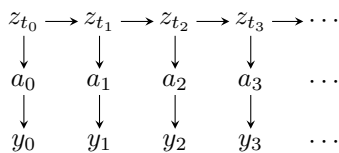
Variant of S4 with an extra latent variable.

Simple hardware-efficient long convolutions for sequence modeling
D.Y. Fu et al. (2023)

The *butterfly decomposition* decomposes the FFT on $n = n_1 n_2$ values into FFTs on n_1 and n_2 values (by reshaping the input as an $n_1 \times n_2$ matrix and applying the FFT on the rows and columns – you can also fine-tune the FFT coefficients, keeping the same sparsity patterns, to get an FFT-like transformation). To improve the performance of the long convolution (S4), tweak the kernel: dropout, smoothing (moving average), quashing of small values.

Neural continuous-discrete state space models for irregularly-sampled time series
A.F. Ansari et al.

If the observations, in a state space model (SSM) (observed at irregular intervals) are high-dimensional, use lower-dimensional auxilliary variables.



Use linear or locally linear dynamics, which interpolate

between SSMs.

$$\begin{aligned} dz &= f(z, t)dt + G(z, t)dB \\ f(z, t) &= F(z)z \\ G(z, t) &= I \\ F(z) &= \sum \alpha_j(z)F_j \\ \alpha(z) &= \text{softmax}(g(z)) \\ g &: \text{neural net} \\ F_j &: \text{basis matrices} \end{aligned}$$

Feature programming for multivariate time series prediction
A. Reneau et al. (2023)

Automatically generate features for multivariate time series by combining operators $\text{Diff}(x_i, x_j) = x_i - x_j$, Lag , Δ , Δ^2 , MA .

The Ising model

$$\begin{aligned} P(\sigma) &\propto \exp -\beta \left(\sum_{(i,j) \in E} J_{ij} \sigma_i \sigma_j + \sum_{i \in V} h_i \sigma_i \right) \\ \sigma_i &\in \{\pm 1\} \end{aligned}$$

can be turned into a dynamic process (Glauber dynamics).

$$\begin{aligned} P(\sigma_{i,t+1}) &\propto \exp(\sigma_{i,t+1} \gamma_{it}) \\ \gamma_{it} &= \sum_j J_{ij} \sigma_j + h_i \end{aligned}$$

Learning perturbations to explain time series predictions
J. Enguehard (2023)

To explain a model f , one can use: a local linear approximation, the gradient of the output wrt the features, or a perturbation: modify the input on a mask, as small/large as possible, to decrease/preserve the performance as much as possible. Instead of a fixed perturbation (average, moving average, Gaussian blur, etc.) use a neural network:

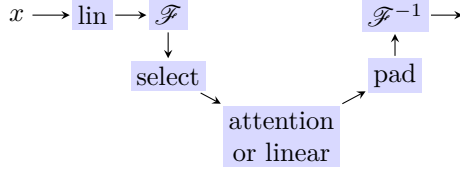
$$\begin{aligned} \text{Find} & \quad m, \text{NN} \\ \text{To minimize} & \quad d[f(x), f(\Phi(x, m))] + \|m\|_1 + \|\text{NN}(x)\|_1 \\ \text{Such that} & \quad 0 \leq m \leq 1 \\ \text{Where} & \quad \Phi(x, m) = m \odot x + (1 - m) \odot \text{NN}(x) \end{aligned}$$

FaDIn: fast discretized inference for Hawkes processes with general parametric kernels
G. Staerman et al. (2023)

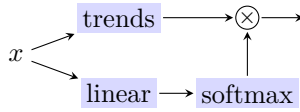
To speed up the estimation of multivariate Hawkes processes with parametric (non-exponential) kernels (which is quadratic in general), assume the kernel has bounded support (limit on the influence of a past event) and discretize the temporal point process.

FEDformer: frequency-enhanced decomposed transformer for long-term series forecasting
T. Zhou et al.

Use a transformer in Fourier (or wavelet) space, but only keep a random subset of the Fourier components. For the wavelets, process the different scales separately.



To extract the trend, use average filters of different sizes, and combine them.



Self-interpretable time series prediction with counterfactual explanations
J. Yan and H. Wang

CountTS is a variational Bayes model for time series tasks (classification, regression, etc.).

Data generation:

$$\begin{aligned} u &\sim N(0, I) \\ z &\sim N(\mu(u, x), \sigma^2(u, x)) \\ y &\sim N(\mu(u, z), \sigma^2(u, z)) \end{aligned}$$

Inference:

$$\begin{aligned} y &\sim x \\ u &\sim x, y \\ z &\sim x, y \end{aligned}$$

Feasible counterfactual explanations only change non-constant parts of the input (e.g., they do not suggest to change the age of the patient).

Sequential multi-dimensional self-supervised learning for clinical time series
A. Raghu et al. (2023)

Use both a global and a cross-sectional self-supervised learning (SSL) loss, e.g, NX-Xent (SimCLR)

$$\text{Mean}_i - \log \frac{\exp(\text{sim}_{ij}/\gamma)}{\sum_{k \neq i} \exp(\text{sim}_{ik}/\gamma)}$$

or VICReg.

Modeling temporal data as continuous functions with stochastic process diffusion
M. Biloš et al. (2023)

DDPM (denoising diffusion probabilistic model) for time series in function space, with noise given by

- A Gaussian process (GP) with RBF kernel $\kappa(\tau) = \exp(-\gamma\tau^2)$;
- An OR diffusion

$$d\varepsilon_t = -\gamma\varepsilon_t dt + dW_t,$$

sampled as a time-changed, scaled Wiener process $\exp(-\gamma t)W_{\exp(2\gamma t)}$, i.e., a GP with a Matérn kernel, $\nu = 1/2$, $\kappa(\tau) = \exp -\gamma |\tau|$.

Applications: forecasting and imputation, for irregular time series.

Probabilistic imputation for time series classification with missing data

VAE-like model for multiple imputation in time series, in the MNAR setup.

Sequential Monte Carlo learning for time series structure discovery
F.A. Saad et al. (2023)

Particle filter (SMC) to select a GP kernel, from a probabilistic context-free grammar (PCFG)

$$\begin{aligned} B &= \text{Linear} \mid \text{Periodic} \mid \text{Gamma} \mid \dots \\ \oplus &= + \mid \times \mid \text{ChangePoint} \\ K &= B \mid (K_1 \oplus K_2) \end{aligned}$$

to model a univariate time series.

I²SB: Image-to-image Schrödinger bridge
G.H. Liu et al.

The Schrödinger bridge generalizes diffusion models.

Provably convergent Schrödinger bridge with applications to probabilistic time series imputation
Y. Chen et al.

The Schrödinger bridge from measure μ to measure ν wrt the prior stochastic process q is the stochastic process

$$p = \underset{\substack{p_0=\mu \\ p_1=\nu}}{\text{Argmin}} \text{KL}(p||q).$$

It generalizes diffusions.

$$\begin{aligned} dx_t &= f(x_t, t)dt + g(t)dW_t \\ dx_t &= [f(x_t, t) - g(t)^2 \nabla \log p_t(x_t)]dt + g(t)d\bar{W}_t \\ dx_t &= [f(x_t, t) - g(t)^2 \nabla \log \vec{\psi}(x_t)]dt + g(t)dW_t \\ dx_t &= [f(x_t, t) + g(t)^2 \nabla \log \vec{\psi}(x_t)]dt + g(t)d\bar{W}_t \end{aligned}$$

**Regions of reliability in the evaluation
of multivariate probabilistic forecasts**
É. Marcotte et al.

A *scoring rule* measures if an observation $y \sim \mathcal{D}$ is consistent with a forecasted distribution $\hat{\mathcal{D}}$; for instance,

$$\text{NLL}(y, \hat{\mathcal{D}}) = -\log p_{\hat{\mathcal{D}}}(y)$$

$$\text{ES}(y, \hat{\mathcal{D}}) = \mathbb{E}_{x \sim \hat{\mathcal{D}}} \|y - x\|^2 - \frac{1}{2} \mathbb{E}_{x, x' \sim \hat{\mathcal{D}}} \|x - x'\|^2$$

$$\text{CRPS}(y, \hat{\mathcal{D}}) = \int_{-\infty}^{+\infty} (\Phi_{\hat{\mathcal{D}}}(x) - \mathbf{1}_{x \geq y})^2 dx$$

$$\text{VG}(y, \hat{\mathcal{D}}) = \sum_{a,b} \left(|y_a - y_b|^p - \mathbb{E}_{x \sim \hat{\mathcal{D}}} |x_a - x_b|^p \right)^2$$

$$\text{DS}(y, \hat{\mathcal{D}}) = \log |\det \Sigma_{\hat{\mathcal{D}}}| + (y - \mu_{\hat{\mathcal{D}}})' \Sigma_{\hat{\mathcal{D}}}^{-1} (y - \mu_{\hat{\mathcal{D}}}).$$

A scoring rule S is *proper* if

$$\forall \hat{\mathcal{D}} \quad \mathbb{E}_{y \sim \mathcal{D}} S(y, \mathcal{D}) \leq \mathbb{E}_{y \sim \mathcal{D}} S(y, \hat{\mathcal{D}}).$$

With small samples, they may fail to spot forecasts with significant imperfections.

**Learning the dynamics
of sparsely observed interacting systems**
L. Bleistein et al. (2023)

Given a controlled differential equation (CDE) $dy_t = G(y_t, x_t)dt$, its solution map $\Psi : (x_{[0,t]}, t) \mapsto y_t$ is linear in signature space: $\Psi(x_{[0,t]}, t) = S_N(x_{[0,t]})'\theta$. The signature transform allows x and y to be irregularly observed time series; y can be sparser.

**Sequential predictive
conformal inference for time series**
C. Xu and Y. Xie (2023)

Conformal inference assumes the data is exchangeable: this assumption does not hold for time series. One could adjust the significance level α to recover the desired coverage (AdaptCI), or leverage the serial dependence of the non-conformity scores with (random forest) quantile regressions (SPCI).

**Resurrecting recurrent neural networks
for long sequences**
A. Orvieto et al. (2023)

Stacking linear recurrent units (LRU – no non-linearity)

$$x_t = \text{diag}(\lambda)x_{t-1} + \gamma \odot u_t$$

$$\lambda_j = \exp(-e^{\nu_j} + ie^{\theta_j})$$


$$\gamma_j = \sqrt{1 - \lambda_j^2}$$

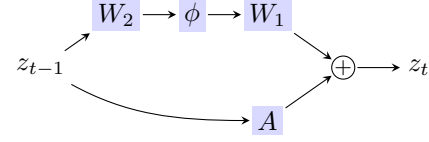
$$u = \text{input}$$

and non-linear MLP blocks recovers the long-range performance of S4.

**Generalized teacher forcing
for learning chaotic dynamics**
F. Hess et al. (2023)

To model (chaotic) dynamical systems and avoid exploding gradients, use:

- Generalized teacher forcing, $\tilde{z}_t = (1 - \alpha)z_t + \alpha\hat{z}_t$, where z_t is from the model and \hat{z}_t is observed;
- A 1-layer RNN, with skip-connection, and clipped ReLU 



**D-CODE: discovering closed form ODEs
from observed trajectories**
Z. Qian et al.

To find an ODE $\dot{x} = f(x)$ from trajectories x , when the data is noisy and/or infrequently sampled (so that we cannot reliably estimate \dot{x} and use symbolic regression, **gplearn**), use the *variational formulation* of $\dot{x} = f(x)$ (multiply by a test function g such that $g(0) = g(T) = 0$, integrate between 0 and T , integrate by parts to remove \dot{x}):

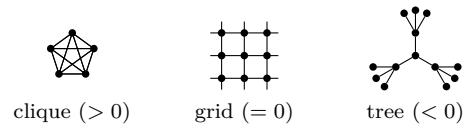
$$C = \int_0^T x(t)\dot{g}(t)dt + \int_0^T f(x(t))g(t)dr.$$

**Understanding over-squashing and bottlenecks
on graphs via curvature**
J. Topping et al.

The *balanced Forman curvature* of an edge $i-j$ counts:

- The number of triangles containing that edge (positive curvature);
- The number of 4-cycles with no diagonal, containing that edge (zero curvature);
- The number of remaining outgoing edges (negative curvature).

Oversquashing is caused by negatively curved edges (bottlenecks).



To limit it, rewire the graph using the stochastic discrete Ricci flow (SDRF):

- Pick the edge $i-j$ with the minimum curvature;
- Add the edge $k-\ell$ increasing the curvature the most, where k and ℓ are neighbours of i and j (or i and j themselves);
- Remove edges with curvature above some threshold.

The *Ollivier curvature* of an edge $i-j$ is

$$\kappa(i, j) = \lim_{\alpha \rightarrow 1} \frac{1 - W_1(\mu_i^\alpha, \mu_j^\alpha)}{1 - \alpha}$$

where

- μ_i^α is the measure on $B_1(i)$ (the ball of radius 1 centered on i , *i.e.*, i and its immediate neighbours) putting weight α on i and $(1 - \alpha)/d_i$ on its neighbours;
- W_1 is the Wasserstein distance (for the geodesic distance on the graph)

$$W_1(\mu_i, \mu_j) = \inf_{\sum_k M_{kw} = \mu_j(w)} \sum_{k,w} M_{kw} d(k, w)$$

The augmented *Forman curvature* only considers triangles.

$$F(i, j) = 4 - d_i - d_j + 3\#_{\Delta}^{ij}$$

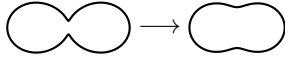
Rewiring networks for graph neural network training using discrete geometry
J. Bober et al.

Same algorithm, but the edge added is picked at random, using the softmax of the curvature improvement.

The *Ricci flow* on a manifold

$$\frac{\partial g}{\partial t} = -2\text{Ric}(g)$$

“smooths” a manifold:



In dimension 2, $\text{Ric}(g) = kg$, where k is the Gauss curvature.

Diffusion improves graph learning
J. Gasteiger et al. (2019)

The generalized diffusion matrix is

$$S = \sum_{k \geq 0} \theta_k T^k,$$

for instance PPR (personalized page rank), $T = AD^{-1}$, $\theta_k = \alpha(1 - \alpha)^k$, or heat diffusion, $T = AD^{-1}$, $\theta_k = e^{-t} t^k / k!$ (add weighted self-loops to the adjacency matrix A ; you can also use the symmetric transition matrix $T_{\text{sym}} = D^{-1/2} A D^{-1/2}$ instead of the random walk one $T_{\text{rw}} = AD^{-1}$).

Truncate it (keeping the top k entries in each column, or the entries above some threshold) and use the resulting (weighted) graph instead of the original one (graph diffusion convolution, GDC, DIGL).

DRew: dynamically rewired message passing with delay
B. Gutteridge et al. (2023)

To limit oversquashing in GNNs, have each node at layer k directly receive input from nodes k hops away (instead of via its immediate neighbours) either their current (DRew) or initial (ν DRew) state.

Expander graph propagation
A. Deac et al. (2022)

The *Cheeger constant* of a connected graph $G = (V, E)$ is

$$h(G) = \min_{\substack{A \subset V \\ 0 < |A| < \frac{1}{2}|V|}} \frac{|\partial A|}{|A|}.$$

Let $0 = \lambda_0 < \lambda_1 < \lambda_2 < \dots$ be the eigenvalues of the Laplacian of a graph G .

An *expander family* is a family of graphs $(G_i)_{i \in I}$ such that $\forall i \lambda_1(G_i) \geq c$, for some $c > 0$; equivalently, $\forall i h(G_i) \geq \varepsilon$, for some $\varepsilon > 0$. In an expander family with bounded degree,

$$\text{diam}(G_i) = O(\log |V_i|).$$

The *Cayley graph* of a group G wrt a generating set S has the elements of G as nodes, and edges $g \rightarrow gs$ for $g \in G, s \in S$.

$$G_n = \text{Cayley} \left(\text{SL}_2(\mathbf{Z}/n), \left\{ \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \right\} \right)$$

is an expander family with $O(n^3)$ nodes. G_n is locally tree-like, up to distance $O(\log n)$.

All the edges have negative curvature, but not enough to produce oversquashing – in any case, expansion, sparsity and non-negative curvature are incompatible: if there is a bound on the degree and the Cheeger constant, there are only finitely many such graphs.

To address oversquashing, interleave a GNN on the input graph with a GNN on the Cayley graph (build it breadth-first, and only keep the first $|V|$ nodes – it is connected).

Other ways of addressing oversquashing include:

- Adding a master node;
- Feature augmentation;
- Graph rewiring (personalized page-rank, stochastic discrete Ricci flow).

Relevant walk search for explaining graph neural networks
P. Xiong et al. (2023)

With the max-product algorithm, layerwise relevance propagation (LRP) can return the top- k most relevant walks in polynomial time.

Brauer’s group equivariant neural network
E. Pearce-Crump (2023)

Description of a generating set of

$$\text{hom}_G((\mathbf{R}^n)^{\otimes k}, (\mathbf{R}^n)^{\otimes \ell})$$

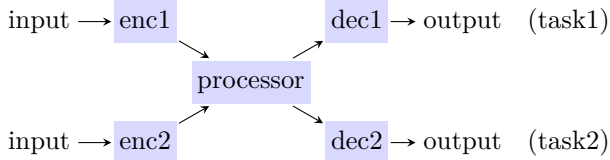
for $G = \text{O}(n)$, $\text{SO}(n)$ or $\text{Sp}(n)$, where the action of G on $(\mathbf{R}^n)^{\otimes k}$ is

$$g \cdot (x_1 \otimes \dots \otimes x_k) = (gx_1) \otimes \dots \otimes (gx_k).$$

In an equivariant neural net, the layers are representations (rather than just vector spaces) and the maps between them are equivariant (representation morphisms).

Neural execution of graph algorithms
P. Veličković et al.

Train a graph neural net to learn, simultaneously, several graph algorithms (e.g., breadth-first search and Bellman-Ford shortest paths) with a shared “processor” and task-specific encoders and decoders; reuse this processor for other graph tasks.



The *Prüfer sequence* of a tree with nodes labeled 1 to n is obtained as follows: in step i , remove the leaf with the smallest label and store the label of its neighbour in position i .

Neural priority queues for graph neural networks
R. Jain et al. (2023)

Differentiable priority queue.

Recursive algorithmic reasoning
D. Jayalath et al.

Augment GNNs with a stack, for recursive tasks, e.g., depth-first search.

Learning to boost training by periodic nowcasting near future weights
J. Jang et al. (2023)

The weight nowcaster network forecasts future (short-term) weights and allows to intermittently skip a few epochs.

A unifying causal framework for analyzing dataset shift-stable learning algorithms
A. Subbaswamy et al. (2022)

When using a causal graph, pay attention to “unstable edges” – edges whose conditional probability distribution may change.

Breaking the curse of depth in graph convolutional networks via refined initialization strategy
S. Wang et al. (2023)

To check if the initialization of a (graph) neural net is satisfactory, look at forward signal propagation, backward signal propagation, and diversity propagation,

$$\frac{\mathbb{E} \|H^{(L)}\|_F^2}{\mathbb{E} \|X\|_F^2} \quad \mathbb{E} \left\| \frac{\partial L}{\partial W^{(1)}} \right\| \quad \mathbb{E} \frac{\text{Dir } H^{(L)}}{\|H^{(L)}\|_F^2}$$

where the Dirichlet energy is $\text{Dir } H = \text{tr}(H' L H)$ and L is the normalized Laplacian.

We do not want those quantities to vanish or explode as $L \rightarrow \infty$; with traditional initialization schemes (Kaiming, Xavier), they do.

MetaInit and SPoGInit scale the weights, layerwise, to improve propagation (make the signal propagations close to 1, and the diversity propagation large).

Adding residual connections, initialized at (0,1), also helps (ReZero, ReZeroGCN)

Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time
M. Wortsman et al. (2022)

When fine-tuning, instead of selecting the best model after hyperparameter search, or averaging the predictions of all the models (ensembling), average the weights, either of all the models (uniform soup), or by adding the models to the soup only if they improve the test set accuracy (greedy soup).

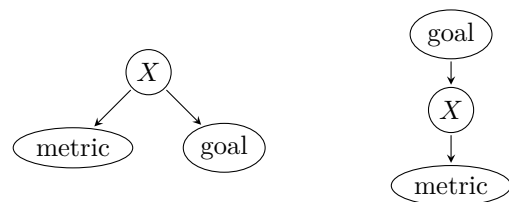
Robust fine-tuning of zero-shot models
M. Wortsman et al.

Fine-tuning improves accuracy on the target distribution, but reduces robustness to distribution shifts. Instead, ensemble the weights of the zero-shot (not fine-tuned) and fine-tuned models (WiSE-FT).

Categorizing variants of Goodhart’s law
D. Manheim and S. Garrabrant (2019)

There are several categories of Goodhart’s law:

- The metric is a noisy version of the intended goal – the optimization may be misled by the noise;
- The metric is a good approximation of the intended goal, but only in a small region of the search space – the optimizer may leave that region;
- The causal relation between the metric and the intended goal is not direct, and someone (the regulator, or the agent – in particular if his own goal is different (cobra effect)) manipulates X .



TRAK: Attributing model behaviour at scale
S.M. Park et al. (2023)

Data attribution decomposes the output of a model $z \mapsto f(z, \theta)$ into a sum of contributions of the training samples S .

$$f(x, \theta^*(S')) \approx \tau(z, S) \cdot \mathbf{1}_{S'} \quad \text{for } S' \subset S$$

This can be done with a lasso linear regression (data-model)

$$\tau(z, S) = \underset{\beta \in \mathbf{R}^n}{\text{Argmin}} \sum_i [\beta' \mathbf{1}_{S_i} - f(z, \theta^*(S_i))]^2 + \lambda \|\beta\|_1$$

for random subsets $S_1, \dots, S_n \subset S$ of sizes $\alpha |S|$, $\alpha \in S$, but this requires fitting n models (thousands).

Instead:

- Linearize the model;
- Reduce the dimension (random projection)

$$\phi(z_i) = P' \nabla_{\theta} f(z_i, \theta^*(S));$$

- The model has become a logistic regression: we can directly compute the approximate leave-out-one influence;
- Ensemble (over random subsets $S_i \subset S$);
- Soft-threshold.

Eigen memory trees
M. Rucker et al. (2022)

High-dimensional variant of quad-trees:

- The data is stored in the leaves of a binary tree;
- Routing is done by testing if

$$\langle x, \text{node.router} \rangle \geq \text{node.boundary};$$

- When a leaf has become too large, it is split using the first principal component of the points it contains.

Application: memory for episodic replay in reinforcement learning (contextual bandits).

Automated search for conjectures on mathematical constants using analysis of integer sequences
O. Razon et al.

Continued fractions can be written

$$\begin{aligned} c &= a_0 + \frac{b_1}{a_1 + \frac{b_2}{a_2 + \frac{b_3}{a_3 + \dots}}} \\ &= a_0 + \sum_{i=1}^{\infty} \frac{b_i}{a_i} \\ &= a_0 + \frac{b_1}{a_1 +} \frac{b_2}{a_2 +} \frac{b_3}{a_3 +} \dots \end{aligned}$$

In a sign-interlaced continued fraction, $(b_i)_{i \geq 1}$ is periodic and $\forall i \ b_i \in \{\pm 1\}$.

The non-terminating Euclidean algorithm for $(\alpha, 1)$ computes the simple ($\forall i \ b_i = 1$) continued fraction of α . It can be extended to allow for sign variation ($\forall i \ b_i \in \{\pm 1\}$), *i.e.*, negative remainders.

The Berlekamp-Massey algorithm finds the minimal linear recurrence sequence with integer coefficients producing a given integer sequence.

Given a constant α (e.g., $\tan 1$), generate its sign-interlaced continued fractions (for all sign patterns up to period T) and look for a linear recurrence for the a_i 's.

Refining generative process with discriminator guidance in score-based diffusion models
D. Kim et al. (2023)

Given a pre-trained diffusion model, train a discriminator $d_{\phi}(x, t)$ to distinguish between real images and generated ones, at all noise levels t . Adjust the learned score function by adding

$$\nabla \log \frac{d_{\phi}(x_t, t)}{1 - d_{\phi}(x_t, t)}.$$

Understanding deep generative models with generalized empirical likelihoods
S. Ravuri et al.

Given n independent samples $x_1, \dots, x_n \sim p$, the *empirical likelihood* method checks if their mean is $c \in \mathbf{R}^d$

$$\underset{x \sim p}{\text{E}} [x] \stackrel{?}{=} c$$

using the weighted empirical distribution $\hat{p}_{\pi} = \sum \pi_i \delta_{x_i}$ maximizing $\text{KL}(\hat{p}_{\pi} \| p)$ with the prescribed mean, where $\hat{p}_n = \sum_i \frac{1}{n} \delta_{x_i}$.

$$\begin{array}{ll} \text{Find} & \pi \in \mathbf{R}^n \\ \text{To maximize} & \sum_i \log \pi_i \\ \text{Such that} & \forall i \ \pi_i \geq 0 \\ & \sum \pi_i = 1 \\ & \sum \pi_i x_i = c \end{array}$$

The *generalized empirical likelihood* uses a moment condition instead,

$$\underset{x \sim p_{\pi}}{\text{E}} [m(x, c)] = 0$$

and a Cressie-Read divergence instead of the KL divergence.

Such divergences include

$$\begin{array}{ll} \text{KL}(\hat{p}_n \| p_{\pi}) & \text{Maximize } \prod \pi_i \\ \text{KL}(p_{\pi} \| \hat{p}_n) & \text{Maximize } - \sum \pi_i \log \pi_i \\ \frac{1}{2} \sum \left(\pi_i - \frac{1}{n} \right)^2 & \text{Minimize } \left(\pi_i - \frac{1}{n} \right)^2. \end{array}$$

Moments include

$$\begin{array}{ll} x - c & \text{mean} \\ \nabla_{\theta} \log p_{\theta}(x) & \text{score function} \\ [\phi(x), \phi(x) \phi(x)'] - c & \text{Fréchet inception distance} \\ k(x, t_i) - k(y, t_i) & \text{mean embedding} \end{array}$$

The mean embedding, which minimizes

$$\sum_i \left[\mathbb{E}_{x \sim p} k(x, t_i) - \mathbb{E}_{x \sim q} k(x, t_i) \right]^2$$

for “witness points” $t_i \sim r$, for some distribution r , is related to the *maximum mean discrepancy* (MMD)

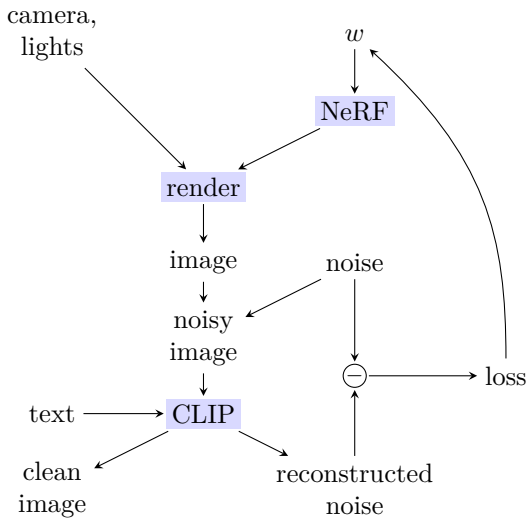
$$\mathbb{E}_{x_1, x_2 \sim p} [k(x_1, x_2)] + \mathbb{E}_{y_1, y_2 \sim q} [k(y_1, y_2)] - 2 \mathbb{E}_{\substack{x \sim p \\ y \sim q}} [k(x, y)].$$

Try the mean embedding, with the exponential kernel $k(x, y) = \exp(x'y/d)$ and the reverse KL divergence (exponential tilting).

With inception, VGG16 or BYOL features ϕ , you can compare images from a generative model with those from the training data, and highlight those unlikely in the other set (e.g., missing modes).

DreamFusion: text-to-3D using 2D diffusion B. Poole et al.

Generate a 3D object as a NeRF (neural radiance field) from text using a pretrained text-to-image diffusion model (CLIP); the model is optimized anew for each new text.



OCD: learning to overfit with conditional diffusion models S. Lutati and L. Wolf (2023)

Diffusion process to generate the weights of another network (hypternetwork); ensemble several of the resulting models.

Evaluating self-supervised learning via risk decomposition Y. Dubois et al. (2023)

For a supervised learning model, the error can be decomposed into that coming from the limited model class and that coming from the finiteness of the training data:

$$\text{test error} = \text{training error} + (\text{test error} - \text{training error}).$$

For self-supervised learning, there are two models, the representation and the probe; the decomposition is finer, accounting for:

- The representation model class, the probe model class;
- Whether the representation can be used by the probe (e.g., linear separability);
- The finiteness of the training data, for the representation and/or the probe.

Efficient self-supervised learning with contextualized target representations for vision, speech and language A. Baevski et al. (2023)

Datavec2.0 is a SimSiam-based multimodal self-supervised learning model.

Unsupervised embedding quality evaluation A. Tsitsulin et al. (2023)

To assess the quality of an unsupervised embedding $M \in \mathbf{R}^{n_1 \times n_2}$:

- α -ReQ: exponent of a power law fit of the singular values, $\lambda_i \propto i^{-\alpha}$ (linear regression on a log-log scale); $\alpha = 1$ suggests good generalization;
- RankMe: effective rank, *i.e.*, entropy of the normalized singular values;
- NESum: flatness of the eigenspectrum of the covariance matrix, $\sum \lambda_i / \lambda_0$; we want the representation to be “whitened”, *i.e.*, $\forall i \lambda \approx \lambda_0$;
- Incoherence: alignment of the singular vectors with the standard basis: $\text{Max}_i \|U'e_i\|$, $\text{Max}_i \|V'e_j\|$;
- Pseudo-condition number $\kappa_2 = \sigma_1 / \sigma_n$;
- Numerical rank: $\|M\|_F / \|M\|_2^2$;
- Clustering (compared with a random distribution on a sphere), from the norm of the pairwise dot product matrix $\|WW'\|_F$.

Self-cluster and coherence give more consistent results.

DataComp: in search of the next generation of multimodal datasets S.Y. Gadre et al.

Multimodal dataset (13b text-image pairs from CommonCrawl, more than ImageNet (1m), CLIP (400m), or StableDiffusion’s LAION-2B (2b)) for *data-centric ML*: how to best filter/preprocess the training data, for a fixed model, training code, and compute budget, for 40 evaluation tasks. The data was sanitized:

- NSFW text filtering: detoxify (multilingual XML-Roberta, threshold 0.1);
- NSFW image filtering (LAION-5B’s CLIP-based NSFW binary classifier, threshold 0.1);
- Deduplication (Yokoo);
- Face blurring (SCRFD face detector)

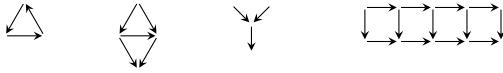
**Neural algorithmic reasoning
with causal regularization
B. Bevilacqua et al. (2023)**

Data augmentation for neural algorithmic reasoning: different inputs leading to the same intermediate computations.

Recall feeds the input back into the model at each step, as a constant reminder of the problem to solve.

**Interventional and counterfactual inference
with diffusion models
P. Chao et al.**

Estimate a structural causal model (with known DAG)) using a diffusion (DDIM) for each node. Experiment using simulated data (e.g.,



10-node random DAG, additive noise, non-additive noise) or real data (2-node graph $\bullet \rightarrow \bullet$).

**Causality inspired representation learning
for domain generalization
F. Lv et al.**

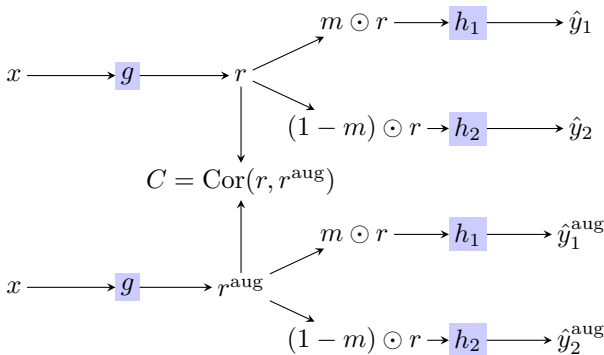
Define augmentations of an image in Fourier space, by keeping the phase, and mixing the amplitude with that of another image.

$$\mathcal{F}(x) = A(x)e^{-iP(x)}$$

$$A(x^{\text{aug}}) = (1 - \lambda)A(x) + \lambda A(x').$$

Learn latent features r such that $\text{Cor}(r, r^{\text{aug}}) \approx I$.

Learn a mask $m = \text{GumbelSoftmax}(w)$ to distinguish between relevant (“causal”) and irrelevant features: the classifiers try to have good performance on both $m \odot r$ and $(1 - m) \odot r$, while the mask tries to have good performance on $m \odot r$ and bad performance on $(1 - m) \odot r$.



$$\mathcal{L}_c = \|C - I\|^2$$

$$\mathcal{L}_{\text{sup}} = \ell(\hat{y}_1, y) + \ell(\hat{y}_1^{\text{avg}}, y)$$

$$\mathcal{L}_{\text{inf}} = \ell(\hat{y}_2, y) + \ell(\hat{y}_2^{\text{avg}}, y)$$

$$\text{Minimize}_{g, f_1, f_2} \mathcal{L}_c + \mathcal{L}_{\text{sup}} + \mathcal{L}_{\text{inf}}$$

$$\text{Maximize}_w \mathcal{L}_{\text{sup}} - \mathcal{L}_{\text{inf}}$$

The *Gumbel softmax* is an approximate, differentiable k -hot vector of size N ; it can be computed as follows.

$$\tau = 0.5$$

$$N : \text{ size of the vector}$$

$$k : \text{ desired number of 1's}$$

$$p \in \Delta_N \text{ probability vector (to be learned)}$$

$$U_{ij} \sim \text{Unif}(0, 1)$$

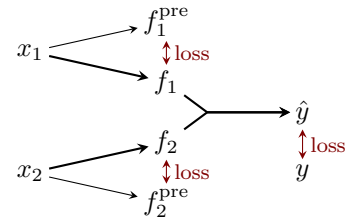
$$\xi_{ij} = -\log(-\log U_{ij}) \text{ Gumbel}$$

$$m_j = \text{Max}_i \frac{\exp \frac{\log p_j + \xi_{ij}}{\tau}}{\sum_{j'} \exp \frac{\log p_{j'} + \xi_{ij'}}{\tau}}$$

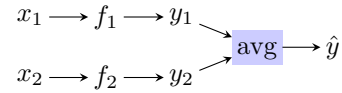
**On unimodal feature learning
in supervised multimodal learning
C. Du et al. (2023)**

To fit a model on multimodal data (e.g., image and text), compare

- Distillation of pretrained unimodal models



- Average of unimodal models



**Fast Poisson disk sampling
in arbitrary dimensions
R. Bridson**

Poisson disk distributions (samples are at least at distance r from one another) are an example of *blue noise*. To generate such samples:

- Keep track of the locations of the points in a grid, with cell size r/\sqrt{n} , where n is the dimension, so that there is at most one point per cell;
- Add a first point, at random, to the active list (and to the grid);
- Pick a point at random in the active list, and generate 30 random points in the spherical annulus of radii r and $2r$ around it; if one of those points is at distance at least r from all the other points (you only need to check adjacent cells), add it to the active list (and to the grid), if not, remove the point from the active list.

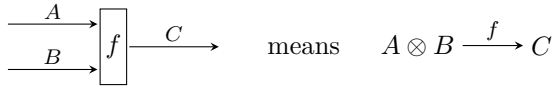
[The centroidal Voronoi tessellation computed from uniformly distributed points also gives blue noise.]

Scientific visualization: Python and Matplotlib
N. Rougier (2021)

How to do everything you thought was impossible with Matplotlib.

A survey of graphical languages for monoidal categories
P. Selinger (2009)

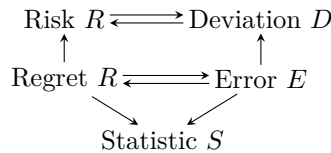
String diagrams (objects and morphisms represented as wires and boxes) for *many* monoidal categories.



Expectile risk quadrangle and applications
V. Kuzmenko et al. (2023)

A *quadrangle* associated to a statistic is a set of 4 functionals, e.g.,

Statistic	$S(x) = \text{VaR}_\alpha(X)$
Risk	$R(X) = \text{CVaR}_\alpha(S)$
Deviation	$D(X) = \text{CVaR}_\alpha[X - EX]$
Regret	$V(X) = (1 - \alpha)^{-1} E[X^+]$
Error	$E(X) = E \left[\frac{\alpha}{1 - \alpha} X^+ + X^- \right]$



One can define quadrangles (they are not unique) for the *expectile*

$$e_q(X) = \underset{c \in \mathbf{R}}{\text{Argmin}} E[X - c]$$

where the asymmetric quadratic error is

$$E_q[X] = q E[(X^+)^2] + (1 - q) E[(X^-)^2].$$

It is the solution of (first order condition)

$$q E[(X - c)^+] = (1 - q) E[(X - c)^-].$$

This leads to estimators of expectile as solutions of linear programs.

The fundamental risk quadrangle in risk management, optimization and statistical estimation
R.T. Rockafellar and S. Uryasev (2013)

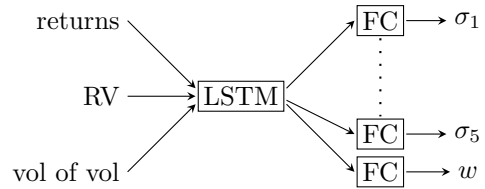
More examples (e.g., S = mean, D = standard deviation, or S = mean, D = variance).

Constructing time-series momentum portfolios with deep multi-task learning
J. Ong and D. Herremans (2023)

Train a neural network fo forecast:

- 21-day forward volatility (5 estimators) using negative correlation as loss (auxiliary tasks);
- Portfolio weights, to maximize the Sharpe ratio

using, as inputs, past log-returns (15, 21, 63, 126, 252), past realized volatility, past vol-of-vol.



Latent factor analysis in short panels
A.P. Fortin et al. (2023)

Test for the number of factors using the norm and eigenvalues of \hat{S} ,

$$\hat{S} = \Sigma^{-1/2} M(V - \Sigma) M' \Sigma^{-1/2}$$

$$V = \frac{1}{n} Y Y' \quad (Y \text{ centered})$$

$$\Sigma = \frac{1}{n} \varepsilon \varepsilon'$$

where the factor model is

$$y_i = \mu + F \beta_i + \varepsilon_i.$$

It can be estimated using

- $\text{diag } V = \text{diag}(F F' + \Sigma)$;
- F contains the first k eigenvectors of $V \Sigma^{-1}$, with eigenvalues $1 + \hat{\gamma}_j$, normalized so that $F' \Sigma^{-1} F = \text{diag}(\hat{\gamma}_1, \dots, \hat{\gamma}_k)$.

A simple method for predicting covariance matrices of financial returns
K. Johansson et al. (2023)

The iterated EWMA estimator of the covariance matrix combines an EWMA for the volatilities and one for the correlation matrix, with a smaller half-life for the former. Combine (the Choleski factors of the precision matrices of) several such estimators for different pairs of half-lives (or other estimators) to maximize the log-likelihood – the weights are the solution of a convex optimization problem (implementation: `cvxgrp/cov_pred_finance`).

Use the regret (difference in log-likelihood with the best (in-sample) constant variance matrix) or the performance of some portfolio (minimum variance, maximum diversity, risk parity, etc.) to compare estimators.

Prediction when factors are weak
S. Giglio et al. (2023)

Supervised PCA (SPCA) selects a subset of predictors, correlated with the target, before computing PCA. A variant proceeds one principal component at a time, projecting predictors and target along it each time.

Mining the factor zoo: estimation of latent factor models with sufficient proxies
R. Wan et al. (2022)

Estimate a time series factor model from a large number of proxy time series

$$\begin{aligned} y_t &= Af_t + \varepsilon_t \\ f_t &= B'x_t + u_t \end{aligned}$$

as (I have left out the scaling factor NK^{-1})

$$\begin{aligned} \hat{A}, \hat{B} &= \underset{\substack{A, B \\ A'A = I}}{\text{Argmin}} \|Y - XBA'\|^2 + \lambda \|B\|^2 \\ \hat{f} &= \hat{A}'y_t. \end{aligned}$$

Non-parametric online market regime detection and regime clustering for multidimensional and path-dependent data structures
B. Horvath and Z. Issa (2023)

The maximum mean discrepancy (MMD) between probability distributions p, q on a topological space X is

$$\begin{aligned} d(p, q) &= \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim p} [f(x)] - \mathbb{E}_{y \sim q} [f(y)] \\ \hat{d}(p, q) &= \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_i f(x_i) - \frac{1}{m} \sum_j f(y_j) \end{aligned}$$

where \mathcal{F} is a set of functions $X \rightarrow \mathbf{R}$, for instance, the unit ball of a RKHS – this is then a distance, and

$$\begin{aligned} \hat{d}(p, q) &= \frac{1}{n(n-1)} \sum_{i \neq j} \kappa(x_i, x_j) - \frac{2}{mn} \sum_{i, j} \kappa(x_i, y_j) + \\ &+ \frac{1}{m(m-1)} \sum_{i \neq j} \kappa(y_i, y_j). \end{aligned}$$

A RKHS on a set X is the datum of:

- A vector space \mathcal{H} of function $X \rightarrow \mathbf{R}$;
- A scalar product $\langle \cdot, \cdot \rangle$ on \mathcal{H} ;
- A positive definite function $\kappa : X \times X \rightarrow \mathbf{R}$ such that:
 - $(\mathcal{H}, \langle \cdot, \cdot \rangle)$ is a Hilbert space;
 - $\forall x \in X \ \kappa(\cdot, x) \in \mathcal{H}$
 - $\forall x \in X \ \forall f \in \mathcal{H} \ \langle f, \kappa(\cdot, x) \rangle = f(x)$.

The *signature kernel* is

$$\kappa(x, y) = \langle S(x), S(y) \rangle$$

where $S(x) \in T((V))$ is the signature transform of $x : [0, 1] \rightarrow V$. It is the solution $\kappa(x, y) = f(1, 1)$ of the hyperbolic PDE

$$f(s, t) = 1 + \int_0^s \int_0^t f(u, v) \langle dx_u, dy_v \rangle_V$$

(there is no need to truncate the signature).

The *kernel scoring rule* compares an observation y with a probability distribution

$$\begin{aligned} s(p, y) &= \mathbb{E}_{x, x' \sim p} \kappa(x, x') - 2 \mathbb{E}_{x \sim p} \kappa(x, y) \\ \hat{s}(p, y) &= \frac{1}{n(n-1)} \sum_{i \neq j} \kappa(x_i, x_j) - \frac{2}{n} \sum_i \kappa(x_i, y) \end{aligned}$$

Apply to changepoint detection, with time-augmented subpaths (normalize the subpaths if needed).

$$\begin{bmatrix} 1 & x_i & x_{i+1} & \cdots & x_{i+h_2-1} \\ 2 & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ h_1 & x_{i+h_1-1} & x_{i+h_1} & \cdots & x_{i+h_1+h_2-2} \end{bmatrix}$$

Risk bounds on aleatoric uncertainty recovery
Y. Zhang et al. (2023)

To estimate the variance in heteroskedastic regression

$$\begin{aligned} Y &= \mu(X) + \varepsilon(X) \\ \mu(X) &= \mathbb{E}[Y|X = x] \\ \sigma^2(x) &= \text{Var}[Y|X = x] \end{aligned}$$

use the Gaussian negative log-likelihood

$$\sum_i \log \sigma^2(x_i) + \frac{(y_i - \mu(x_i))^2}{\sigma_i^2(x_i)}$$

or the generalized method of moments (GMM)

$$\sum [\sigma^2(x_i) - (y - \mu(x_i))^2]^2$$

or a Gaussian process

$$\begin{aligned} g &\sim \text{GP} \quad (\text{prior}) \\ \sigma^2(x) &= \exp g(x). \end{aligned}$$

Never mind the metrics – what about the uncertainty? Visualizing confusion matrix metric distributions
D. Lovell et al.

Exhaustive list of performance metrics; in particular, you should also report balanced accuracy (BA) and the

Matthews correlation coefficient (MCC).

$$\begin{aligned} \text{BA} &= \frac{\text{TPR} + \text{TNR}}{2} \\ \phi = \text{MCC} &= \frac{\text{TP} \cdot \text{TN} - \text{FP} \cdot \text{FN}}{\sqrt{\hat{p}\hat{p}\hat{n}\hat{n}}} \\ p &= \text{TP} + \text{FN} \\ n &= \text{FP} + \text{TN} \\ \hat{p} &= \text{TP} + \text{FP} \\ \hat{n} &= \text{FN} + \text{TN} \\ \text{TPR} &= \text{TP}/p \\ \text{TNR} &= \text{TN}/n \end{aligned}$$

Plot both the ROC and precision-recall curves. Show, in the background, all possible confusion matrices (as a grid) with the same size N and prevalence P/N (proportion of positives), coloured with MCC or BA (you can also add the MCC level curves).

***Fisher-Rao distance
between multivariate normal distributions:
special cases, bounds and applications***
J. Pinele et al. (2020)

There is no known closed formula for the Fisher-Rao distance between Gaussians in general, but there is one in many special cases. For instance, when the mean is the same:

$$d((\mu, \Sigma_1), (\mu, \Sigma_2)) = \sqrt{\frac{1}{2} \sum (\log \lambda_i)^2}$$

where the λ_i are the eigenvalues of $\Sigma_1^{-1/2} \Sigma_2 \Sigma_1^{-1/2}$, *i.e.*, the generalized eigenvalues of (Σ_1, Σ_2) .

***Trust region methods
on Riemannian manifolds***
P.A. Absil et al.

Trust region methods, to minimize $f : \mathbf{R}^n \rightarrow \mathbf{R}$, iteratively solve (e.g., with *truncated conjugate gradient*)

$$\underset{\eta \in \mathbf{R}^n, \|\eta\| \leq \Delta}{\text{Minimize}} \underbrace{f(x) + f'(x)\eta + \frac{1}{2}\eta' f''(x)\eta}_{m(\eta)}$$

updating the radius Δ by looking at the quality of the approximation

$$\rho = \frac{f(x) - f(x + \eta)}{m(0) - m(\eta)},$$

e.g.,

$$\begin{aligned} \rho &\ll 1 && \text{discard the update and decrease } \Delta \\ \rho &< 1 && \text{decrease } \Delta \\ \rho &\approx 1 && \text{increase } \Delta. \end{aligned}$$

It still works if $f''(x)$ is replaced by an approximation of the Hessian.

This generalizes to functions on a Riemannian manifold $f : M \rightarrow \mathbf{R}$, by using the exponential map

$\exp_x : T_x M \rightarrow M$ or (since the exponential may be expensive to compute), a *retraction*

$$\begin{aligned} R_x : T_x M &\rightarrow M \\ R_x(0) &= x \\ DR_x(0) &= \text{Id} \end{aligned}$$

(no condition on the second derivative).

Examples include:

- Minimize $Q \in O_n$ $\text{trace}(Q' A Q N)$, where $N = \text{diag}(1, 2, \dots, n)$ (this gives the eigenvectors of A , sorted) and we can replace $\exp \Omega$ with a product of Givens rotations;
- The extreme eigenspace of (A, B) (the span of the p generalized eigenvectors with the lowest eigenvalues) in $\text{Grass}(p, n) = \mathbf{R}_*^{n \times p} / \text{GL}_p$, which is not naturally a submanifold of \mathbf{R}^N .

***Pymanopt: a Python toolbox for optimization
on manifolds using automatic differentiation***
J. Townsend et al. (2016)

Example: find the best rank- k positive semidefinite approximation of a matrix for the *pseudo-Huber loss* $H(x) = \sqrt{x^2 + \delta^2} - \delta$ (which is quadratic for small x and L^1 for large x) using a Riemannian trust region solver.

Understanding deep learning
S.J.D. Prince (2023)

3. There is a zoo of activation functions:

$$\begin{aligned} \sigma(z) &= (1 + e^{-z})^{-1} \\ \tanh z & \\ \text{PreLU}_a(z) &= \begin{cases} z & \text{if } z \geq 0 \\ az & \text{if } z \leq 0 \end{cases} \\ \text{LReLU} &= \text{PreLU}_a \\ \text{Softplus}(z) &= \log(1 + e^z) \\ \text{GeLU}(z) &= z \cdot \Phi(z) \\ \text{SiLU}(z) &= z \cdot \sigma(z) \\ \text{ELU}_\alpha(z) &= \begin{cases} z & \text{if } z \geq 0 \\ \alpha(e^z - 1) & \text{if } z \leq 0 \end{cases} \\ \text{SELU}(z) &= \lambda \text{ELU}_\alpha(z) \\ \text{Swish}_\beta(z) &= z \cdot \sigma(\beta z) \end{aligned}$$

4. Deep neural networks are easier to train and generalize better than shallow ones.

5. The cross-entropy loss is equal, up to an additive

constant, to the negative log-likelihood.

$$\begin{aligned}\text{KL}(q \parallel p) &= \mathbb{E}_{z \sim q} \left[\log \frac{q(z)}{p(z)} \right] \\ \hat{\theta} &= \underset{\theta}{\text{Argmin}} \text{KL}(\text{data} \parallel \text{model}_{\theta}) \\ &= \underset{\theta}{\text{Argmin}} \mathbb{E}_{z \sim q_{\text{data}}} \left[\log \frac{q_{\text{data}}(z)}{p_{\theta}(z)} \right] \\ &= \underset{\theta}{\text{Argmin}} \mathbb{E}_{z \sim \text{data}} [-\log p_{\theta}(z)]\end{aligned}$$

Many other losses can be used, e.g.,

- Negative log-likelihood from other models;
- Pinball loss (quantile regression);
- Focal loss (downweight well-calssified examples);
- Hinge loss (SVM);
- Exponential loss (AdaBoost);
- Learning-to-rank;
- etc.

6. Adam combines momentum and RMSProp (parameter-specific learning rates); it often makes rapid initial progress, but you may want to switch to SGD after a while. Learning rate warm-up may be needed.

7. To avoid vanishing or exploding gradients, initialize the weights so that the activations have mean 0 and variance 1, e.g., with He initialization

$$\sigma^2 = \frac{4}{d_{\text{in}} + d_{\text{out}}}$$

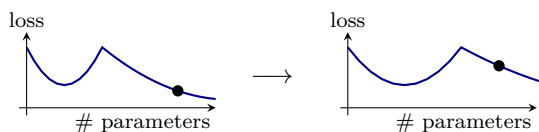
(initially derived for linear layers) or Glorot initialization (multiply by 2, to account for ReLU activation functions).

8. There are three sources of error:

- Noise: the data generation process is stochastic;
- Bias: the data generation process is not in the family of models considered;
- Variance: the model is estimated from a finite sample.

The *double descent* phenomenon depends on the loss, the initialization, the optimization algorithm (which provides implicit regularization); it is more visible with noisy training data.

Adding more data can have a negative impact on performance – the model is less over-parametrized.



Data drift can manifest itself as:

- Change in x : covariate shift;
- Change in y : prior shift;
- Change in $y|x$: concept shift.

9. The discreteness of the time steps in gradient descent are equivalent to an implicit regularization:

$$\begin{aligned}\text{loss}_{\text{GD}}(\phi) &= \text{loss}(\phi) + \frac{\alpha}{4} \left\| \frac{\partial L}{\partial \phi} \right\|^2 \\ \phi_{t+1} &= \phi_t - \alpha \frac{\partial \text{loss}}{\partial \phi},\end{aligned}$$

i.e., the trajectory avoids places with a steep gradient (and the regularization becomes stronger with larger step sizes α).

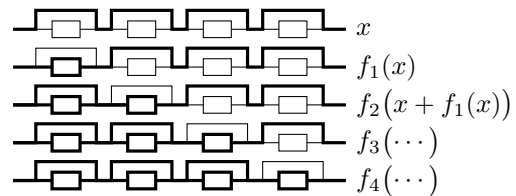
The minibatches in SGD introduce another implicit regularization

$$\frac{\alpha}{4B} \sum_{b=1}^B \left\| \frac{\partial \text{loss}_b}{\partial \phi} - \frac{\partial \text{loss}}{\partial \phi} \right\|^2$$

i.e., the trajectory prefers places where gradients do not differ too much between batches.

To improve performance, try early stopping, ensembling, dropout, noise (in the inputs (adversarial training), the weights, or the labels (label smoothing)), pre-training, augmentation, self-supervised learning.

11. Very deep networks suffer from *shattered gradients*: small changes in the input can lead to large changes in the gradient of the loss. Residual (skip) connections address that problem. A ResNet can be seen as an ensemble of smaller networks:



Batch normalization helps with exploding/vanishing gradients.

Variations of ResNets include:

- UNet;
- Stacked hourglass networks (stacked UNets);
- DenseNet (concatenates the outputs of all previous layers to feed the current layer).

13. An *inductive* model sees labeled data at training time and unlabeled data at test time. A *transductive* model sees both labeled and unlabeled data at training time (semi-supervised learning) – node classification models are often transductive.

Graph attention

$$\begin{aligned}S_{mn} &= a \left[\phi' \begin{pmatrix} h_m \\ h_n \end{pmatrix} \right] \\ H &\leftarrow a [H' \cdot \text{Softmask}(S, A + I)]\end{aligned}$$

differs from traditional attention:

- $Q = K = V = H$;
- There is a mask, $A + I$;
- $S \neq K'Q$.

For large graphs, minibatch training is tricky: try adding a subset of the nodes in the receptive field at each stage, or sample a subgraph with random walks.

14. Unsupervised learning includes

- GANs;
- VAEs;
- Normalizing flows;
- Diffusion models.

The inception score (of an image generation model) is the Fréchet distance between the latent representations (penultimate activations of an inception model) of real and generated data, after Gaussian approximation.

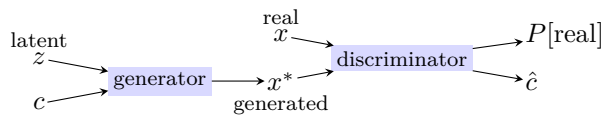
$$\|\mu_X - \mu_Y\|^2 + \text{tr}[\Sigma_X + \Sigma_Y - 2(\Sigma_X \Sigma_Y)^{1/2}]$$

15. GANs are difficult to train and suffer from *mode dropping*, or even *mode collapse*. *Minibatch discrimination* adds a penalty to ensure diversity in each minibatch.

If the discriminator is too good (in particular, if it recognizes that real and generated data are disjoint), the gradients are uninformative and the generator cannot improve. The *Wasserstein loss* does not have that problem: the gradients remain informative.

GAN variants include:

- Conditional GAN: generator and discriminator are given the class of the input;
- Classifier GAN: the generator is given the class, and the discriminator tries to recover it;
- InfoGAN: the discriminator tries to recover part of the noise given to the generator (this helps disentangle the latent space);



- Superresolution GAN;
- CycleGAN;
- StyleGAN: add the latent variable (and the noise) at several points in the network, *i.e.*, at different scales;
- MADGAN: several generators, and the discriminator tries to guess which one was used.

16. Normalizing flows are invertible layers (or networks) whose Jacobians have an easy-to-compute determinant. For linear layers, this is easy to obtain with

the LU decomposition

$$\begin{aligned} f(h) &= \beta + \Omega h \\ \Omega &= PL(U + D) \\ P &\in \mathfrak{S}_n \\ L &= \begin{bmatrix} * & & 0 \\ | & \diagdown & \\ * & \text{---} & * \end{bmatrix} \\ U &= \begin{bmatrix} 0 & & * \\ | & \diagdown & \\ 0 & \text{---} & 0 \end{bmatrix} \\ D &= \begin{bmatrix} * & & 0 \\ | & \diagdown & \\ 0 & & * \end{bmatrix} \end{aligned}$$

They can be combined with an invertible elementwise nonlinearity, e.g., LeakyReLU.

Coupling flows generalize the linear invertible transformation

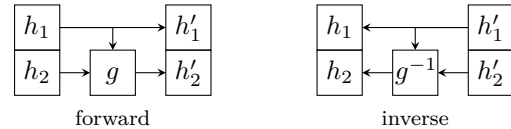
$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 \\ * & 1 \end{pmatrix} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$$

to

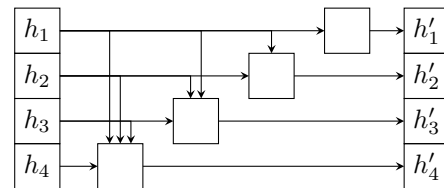
$$\begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \mapsto \begin{pmatrix} h_1 \\ g_{\phi(h_1)} h_2 \end{pmatrix}$$

with inverse

$$\begin{pmatrix} h_1 \\ g_{\phi(h_1)}^{-1} h_2 \end{pmatrix} \longleftarrow \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}$$

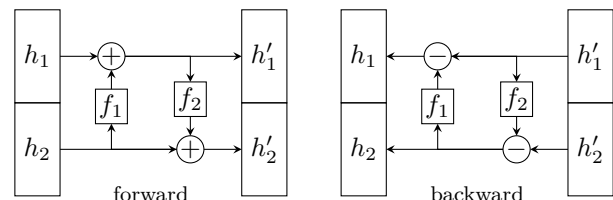


Autoregressive flows generalize this to more blocks.



Residual flows use a similar idea.

$$\begin{aligned} \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} &\mapsto \begin{pmatrix} h'_1 \\ h'_2 \end{pmatrix} = \begin{pmatrix} h_1 + f_1(h_2) \\ h_2 + f_2(h'_1) \end{pmatrix} \\ \begin{pmatrix} h'_1 - f_1(h_2) \\ h'_2 - f_2(h'_1) \end{pmatrix} &\longleftarrow \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \end{aligned}$$



Multiscale flows generalize them to more than two blocks.

17. To model $p(x)$, latent variable models model $p(x, z)$, often as $p(x, z) = p(x|z)p(z)$, and marginalize z : $p(x) = \mathbb{E}_z p(x, z)$. Examples include:

- Gaussian mixture;

$$\begin{aligned} z &\sim \text{Discrete} \\ x|z &\sim N(\mu_z, \sigma_z^2) \end{aligned}$$

- Nonlinear Gaussian mixtures

$$\begin{aligned} z &\sim N(0, I) \\ x|z &\sim N(f_\phi(z), \sigma^2 I) \end{aligned}$$

The expectation $\mathbb{E}_z p(x, z)$ is intractable, but amenable to variational inference.

$$\begin{aligned} \log p(x) &= \log \int p(x, z) dz \\ &= \log \int q(z) \frac{p(x, z)}{q(z)} dz \quad (q \text{ arbitrary}) \\ &\geq \int q(z) \log \frac{p(x, z)}{q(z)} dz \quad (\text{Jensen}) \\ &= \text{ELBO}(q, p) \end{aligned}$$

$$\begin{aligned} \text{ELBO}(q, p) &= \int q(z) \log \frac{p(x, z)}{q(z)} dz \\ &= \int q(z) \log \frac{p(z|x)p(x)}{q(z)} dz \\ &= \int q(z) \log p(x) dz + \int q(z) \log \frac{p(z|x)}{q(z)} dz \\ &= \log p(x) + \text{KL}(q(z) \| p(z|x)) \end{aligned}$$

$$\begin{aligned} \text{ELBO}(q, p) &= \int q(z) \log \frac{p(x, z)}{q(z)} dz \\ &= \int q(z) \log \frac{p(x|z)p(z)}{q(z)} dz \\ &= \int q(z) \log p(x|z) dz + \int q(z) \frac{p(z)}{q(z)} dz \\ &= \int q(z) \log p(x|z) dz + \text{KL}(q(z) \| p(z)) \\ &= \text{reconstruction loss} + \text{distance to prior} \end{aligned}$$

This is the loss used by VAEs.

For the distributions, use

$$\begin{aligned} p_\phi(x|z) &= \text{Norm}_x(f(z, \phi), \sigma^2 I) \\ q_{\theta, x}(z) &= \text{Norm}_z(g_\mu(x, \theta), g_\Sigma(x, \theta)) \\ p(z) &= \text{Norm}_z(0, I). \end{aligned}$$

The first term is intractable, but can be approximated with Monte Carlo, by sampling $z \sim q_{\theta, x}$ (one sample may be good enough). The second term is the KL divergence between two Gaussians,

$$\text{KL}(N(\mu, \Sigma) \| N(0, I)) = \frac{1}{2} (\text{tr } \Sigma + \mu' \mu - n - \log \det \Sigma).$$

The β -VAE rescales the second term (KL penalty)

$$\text{loss} = \log p_\phi(x, z^*) - \beta \cdot \text{KL}(q_{\theta, x}(z) \| p(z))$$

where z^* is a single sample.

18. Diffusion models progressively add noise to an image

$$\begin{aligned} z_0 &= x \\ z_t &= \sqrt{1 - \beta_t} z_{t-1} + \sqrt{\beta_t} \varepsilon_t \end{aligned}$$

and try to reverse this process. If we knew x , we could compute $q(z_{t-1}|z_t, x)$ – it is Gaussian. Approximate $q(z_{t-1}|z_t)$ with Gaussian distributions

$$z_{t-1}|z_t \sim N(f_t(z_t, \phi_t), \sigma_t^2 I)$$

and use variational inference.

StableDiffusion does this in a latent space (smaller than pixel space).

20. It is not clear why deep learning works:

- It is not because the dataset is easy to learn: one can fit random data almost as easily;
- It is not because of (explicit or implicit) regularization;
- It is not because SGD, with small batches, moves easily between valleys – large batches work equally well;
- Activation functions are important: they should provide informative gradients;
- Initialization is only important for deep networks;
- Inductive bias (CNNs for grid data, transformers for \mathfrak{S}_n -invariance) is important;
- Overparametrization and depth are important (lottery ticket hypothesis: a large network can be seen as an ensemble of smaller subnetworks, one of which may have been initialized near a minimum); overparametrization also allows the model to be smoother between data points;
- Regularization can explain double descent and *grokking* (sudden improvement in generalization long after the training error reached zero)

The loss surface has interesting properties:

- There are many global minima; they seem to be connected by low-loss regions;
- The optimization path lies in a low-dimensional subspace;
- There are few or no bad local minima (but there are many saddle points).

A simple explanation of partial least squares K.S. Ng (2013)

The principal components of X (assuming the data is centered) can be computed from the eigen decomposition of $X'X$.

$$\begin{aligned} X'X &= PDP' && \text{eigen decomposition} \\ P &&& \text{principal components} \\ T &= XP && \text{scores} \\ X &= TP' && (\text{because } P^{-1} = P') \end{aligned}$$

Often, we only keep the first k principal components $P_{\leq k}$.

$$T_{\leq k} = XP_{\leq k}$$

The principal components can also be computed from the singular value decomposition (SVD).

$$X = (U\Sigma)P' = TP'$$

The NIPALS algorithm uses power iterations to find the principal components one by one,

```
Repeat
  t = random
  Repeat
    p ∝ X't
    t = Xp
  X ← X - tp'
```

Given two datasets, X and Y , consider the NIPALS algorithms for them, but swap the scores of X and Y , and interleave the updates.

PCA	PLS
Repeat	Repeat
$t = \text{random}$	$t = \text{random}$
$u = \text{random}$	$u = \text{random}$
Repeat	Repeat
$p \propto X't$	$p \propto X'u$
$t = Xp$	$t = Xp$
$q \propto U'u$	$q \propto U't$
$y = Yq$	$y = Yq$
$X \leftarrow X - tp'$	$X \leftarrow X - tp'$
$X \leftarrow Y - uq'$	$X \leftarrow Y - uq'$

This gives decompositions

$$\begin{aligned} X &= TP' \\ Y &= UQ'. \end{aligned}$$

PLS regression estimates a regression between the scores (latent) $U \sim T$, giving $U \approx T\beta$, and uses it to express Y from X :

$$\begin{aligned} Y &= UQ' \\ &\approx T\beta Q' \\ &= XP\beta Q' \\ &= XB_{\text{PLS}}. \end{aligned}$$

For p , this is the power iteration

$$p \propto X'u \propto X'Yq \propto X'YY't \propto X'YY'Xp$$

to compute the largest eigenvalue of $X'YY'X$, i.e.,

$$\begin{aligned} p &= \underset{\|p\|=1}{\text{Argmax}} p'X'YY'Xp \\ &= \underset{\|p\|=1}{\text{Argmax}} (Y'Xp)Y'Xp \\ &= \underset{\|p\|=1}{\text{Argmax}} \text{Cov}(Y, Xp)' \text{Cov}(Y, Xp) \\ &= \underset{\|p\|=1}{\text{Argmax}} \|\text{Cov}(Y, Xp)\|. \end{aligned}$$

Note that p and q are the right and left singular vectors of $X'Y$:

$$p, q = \underset{\|p\|=\|q\|=1}{\text{Argmax}} \text{Cov}(Xp, Xq).$$

A survey of partial least squares (PLS) methods, with emphasis on the 2-block case **J.A. Wegelin (2000)**

There are (at least) 3 variants of the PLS algorithm. They all use a rank-1 approximation of $A = X'Y$ (from the SVD)

$$\begin{aligned} A &\approx dpq' \\ t &= Xp \\ u &= Yq \end{aligned}$$

but differ on how X and Y (or A) are updated:

PLS-W2A	$X \leftarrow X - t(t't)^{-1}t'X$ $Y \leftarrow Y - u(u'u)^{-1}u'Y$
PLS-SVD	$A \leftarrow A - dpq'$
PLS2	$X \leftarrow X - t(t't)^{-1}t'X$ $Y \leftarrow Y - (t't)^{-1}(t'u)tu'$

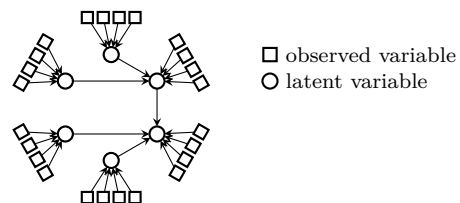
PLS1 is the special case of PLS2 where Y only has one variable.

CCA (canonical correlation analysis) replaces the covariance with the correlation:

$$p, q = \underset{\|p\|\|q\|=1}{\text{Argmax}} \text{Cor}(Xp, Xq).$$

This requires inverting $X'X$ and $Y'Y$, which can lead to numeric instabilities.

PLS can be generalized to more than two blocks (PLS-SEM – not recommended) if we posit the relations between the latent variables (path diagram).



Partial least squares methods: partial least squares correlation and partial least squares regression **H. Abdi and L.J. Williams (2013)**

Confusingly, PLS can refer to two things:

- PLS correlation, finding latent factors common to two datasets;
- PLS regression, predicting one dataset from another.

Given two data matrices $X \in \mathbf{R}^{I \times J}$, $Y \in \mathbf{R}^{I \times K}$ (with centered and normalized columns), PLSC finds the unit vectors $u_1 \in \mathbf{R}^K$, $v_1 \in \mathbf{R}^J$ maximizing the covariance $(Xv_1)'(Yu_1)$, then the unit vectors, orthogonal to u_1 and v_1 , maximizing $(Xv_2)'(Yu_2)$, and so on. Those vectors can be computed in one go from the SVD

$$\begin{aligned} Y'X &= U\Delta V' \\ U &: \text{loadings of } Y \\ V &: \text{loadings of } X \\ XV &: \text{factor scores} \\ YU &: \text{factor scores} \end{aligned}$$

To assess significance, compute the sum of the (retained) singular values (inertia) and compare with that obtained after shuffling the rows of X (bootstrap distribution of the inertia).

PLS regression computes latent factors an X , and regresses Y on those factors:

$$\begin{aligned} X &= TP' \\ \hat{Y} &= YBC' = (P')^\dagger BC'. \end{aligned}$$

It can be computed iteratively:

$$\begin{aligned} X'Y &= W\Delta C' && (\text{SVD}) \\ &\approx w\delta c' && (\text{rank 1}) \\ t &= Xw && X \text{ latent} \\ p &= X't && X \text{ loadings} \\ u &= Yc \\ \hat{X} &= tp' \\ \hat{Y} &= uc' \\ &= tt'uc' \\ &= tbc' \\ b &= t'u && \text{slope} \\ X &\leftarrow X - \hat{X} \\ Y &\leftarrow Y - \hat{Y}. \end{aligned}$$

After $L = \text{rank } X$ iterations:

$$\begin{aligned} X &= TP' \\ \hat{Y} &= TBC' \\ &= X(P')^\dagger BC' \end{aligned}$$

This solves the optimization problem

$$\begin{aligned} \text{Find} & \quad w \\ \text{To maximize} & \quad \|\text{Cov}(Xw, Y)\|^2 \\ \text{Such that} & \quad \|w\| = 1. \end{aligned}$$

Supervised linear dimension reduction methods: review, extensions and comparisons
S. Xu et al.

Supervised dimension reduction methods include:

- PCA on the ℓ variables most correlated with y (choose ℓ to get the best performance when predicting y from the top k principal components, k fixed);
- Idem, but iteratively, extracting one principal component at a time;
- k principal components most correlated with y (instead of the top k);
- PLS-W2A

$$\begin{aligned} u &= \underset{\|u\|=1}{\text{Argmax}} u'Xyy'Xu \\ z &= Xu \\ X &\leftarrow X - zu' \\ y &\leftarrow y - \frac{\langle y, z \rangle}{\|z\|^2} z; \end{aligned}$$

- Extended PLS-W2A, *i.e.*, with a PCA penalty,

$$\underset{\|u\|=1}{\text{Maximize}} u'Xyy'Xu - \lambda \|X - Xu u'\|^2$$

- PLS-SVD

$$\underset{U'U=I}{\text{Maximize}} \|\text{Cov}(XU, y)\|$$

- Extended PLS-SVD

$$\underset{U'U=I}{\text{Maximize}} \|\text{Cov}(XU, y)\| - \lambda \|X - XU U'\|^2$$

- LSPCA (least-squares PCA, no closed-form solution)

$$\underset{\beta, U'U=I}{\text{Minimize}} \|y - XU\beta\|^2 - \lambda \|X - XU U'\|^2$$

- Supervised probabilistic PCA (EM algorithm)

$$\begin{aligned} x &= Uz + \varepsilon \\ y &= v'z + \eta \end{aligned}$$

Prefer PLS-W2A and LSPCA.

Supervised dimension reduction for big data
J.T.Vogelstein et al.

PCA can be adapted to include a target qualitative variable. In the case of two classes:

- Compute the mean of each class;
- The difference between the means is the first component;
- Subtract the mean;
- Perform PCA from the covariance matrix of the centered data.

This can be generalized to more than 2 classes, and to robust alternatives of the mean and variance.

SLISEMAP: supervised dimensionality reduction through local explanations
A. Björklund et al. (2022)

Dimension reduction (supervised UMAP), not on the features, but on LIME explanations.

$$\begin{aligned} z_i &: \text{embedding of observation } i \text{ (unknown)} \\ d_{ij} &= d(z_i, z_j) \\ w_{ij} &\propto e^{d_{ij}} \quad \sum_j w_{ij} = 1 \\ L_{ij} &= \text{loss}(g_i(x_j), y_j) \end{aligned}$$

where L_{ij} is the loss of the local model g_i for observation i , applied to observation j . The latent embeddings are

$$\text{Minimize}_z \sum_{ij} w_{ij} L_{ij} \quad \text{s.t.} \quad \langle \|z_i\|^2 \rangle = r^2.$$

Supervised discriminative PCA with adaptive neighbours for dimensionality reduction
Z. Shi et al.

PCA can be written

$$\text{Maximize}_W \text{tr}(X'X'XW) \quad \text{s.t.} \quad W'W = I_k$$

or

$$\text{Minimize}_W \|X - XWW'\|^2 \quad \text{s.t.} \quad W'W = I_k.$$

The variant

$$\text{Maximize}_Q \text{tr}(Q'XX'Q) \quad \text{s.t.} \quad Q'Q = I_k$$

$$\text{Minimize}_Q \|X - XX'X\|^2 \quad \text{s.t.} \quad W'W = I_k$$

is equivalent if the data has been standardized (the projections are respectively $W_{\text{PCA}} = V_{1:k}$ and $W_{\text{vPCA}} = V_{1:k}\Sigma_{1:k}$, where $X = U\Sigma V'$ is the SVD).

SDSPCA adds a target variable,

$$\text{Minimize}_{W,G,Q} \|X - QW'\|^2 + \alpha \|Y - QG\|^2 + \beta \|Q\|_2 \quad Q'Q = I_k$$

(it can be computed from the eigenvectors of $-XX' - \alpha YY' + \beta D$).

PCAN is a PCA variant preserving neighbourhood information

$$\begin{aligned} \text{Find} \quad & W, F, S \\ \text{To minimize} \quad & \sum_{ij} \|W'x_i - W'x_j\|^2 S_{ij} + \gamma_i S_{ij}^2 + \\ & + \lambda \|f_i - f_j\|^2 S_{ij} \\ \text{Such that} \quad & S\mathbf{1} = \mathbf{1} \\ & W'X'XW = I_k \\ & F'F = I_c \end{aligned}$$

One can combine SDSPCA and PCAN.

Financial and macroeconomic data through the lens of a nonlinear dynamic factor model
P.A. Guerrón et al.

The dynamic factor model

$$\begin{aligned} Y_t &= \Lambda F_t + \varepsilon_t \\ F_t &= \Psi(L)F_{t-1} + \eta_t \end{aligned}$$

can be made non-linear

$$\begin{aligned} y_t &= \mathcal{G}(f_t) + \varepsilon_t \\ f_t &= \mathcal{H}(f_{t-1}) + \eta_t. \end{aligned}$$

The pruned second order state space model

$$\begin{aligned} y_t &= Gf_t + \varepsilon_t \\ f_t &= x + a_t + b_t \\ a_t &= \alpha a - t - 1 + \eta_t \\ b_t &= \beta b_{t-1} + \gamma a_{t-1}^2 \end{aligned}$$

can be estimated with a particle filter (you may need hundreds of thousands of particles).

Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data
C.H. Martin et al.

For each weight matrix W , compute (with **weight-watcher**): the eigenvalues of $W'W$ (they tend to look like the eigenvalues of a random matrix, or the eigenvalues of a random matrix with a few outliers, or a heavy-tail distribution), their power law exponent α , and

$$\begin{aligned} & \log \sum \lambda_i \\ & \log \lambda_{\max} \\ & \alpha \log \lambda_{\max} \\ & \log \sum \lambda_i^\alpha \end{aligned}$$

and aggregate those quantities across layers

$$\begin{aligned} & \sum_\ell \log \sum \lambda_{i\ell} \\ & \sum_\ell \log \lambda_{\max, \ell} \\ & \frac{1}{L} \sum_\ell \alpha_\ell \log \lambda_{\max, \ell} \\ & \sum_\ell \log \sum \lambda_{i\ell}^{\alpha_\ell} \\ & \frac{1}{L} \sum_\ell \alpha_\ell. \end{aligned}$$

Trained networks with good accuracy tend to have lower values of those metrics.

Tree-ring watermarks: fingerprints for diffusion images that are invisible and robust
Y. Wen et al.

Embed the watermark into the initial noise vector, in Fourier space, as rings of constant value.

Neural set function extensions: learning with discrete functions in high dimensions
N. Karalias et al. (2022)

A set function $f : \{0, 1\}^{[n]} \rightarrow \mathbf{R}$ can be extended to

$$\tilde{f} : \begin{cases} [0, 1]^n & \rightarrow \mathbf{R} \\ x & \mapsto \tilde{f}(x) = \sum_{S \subset [n]} p_x(S) f(S) \end{cases}$$

where p_x satisfies

$$\begin{aligned} \sum_{S \subset [n]} p_x(S) \mathbf{1}_S &= x \\ \sum_{S \subset [n]} p_x(S) &= 1 \\ \forall S \subset [n] \quad p_x(S) &\geq 0. \end{aligned}$$

It can further be extended to positive semidefinite matrices, \mathbf{S}_+^n , e.g.,

$$\tilde{f}(X) = \sum_{S, T \subset [n]} \sum_i \lambda_i p_{x_i}(S) p_{x_i}(T) f(S \cap T)$$

where $X = \sum \lambda_i x_i x_i'$ is the eigen decomposition of X and p_x is as above. More generally,

$$\tilde{f}(X) = \sum_{S, T \subset [n]} p_X(S, T) f(S \cap T)$$

where p_X satisfies

$$\begin{aligned} X &\preceq \frac{1}{2} p_X(S, T) (\mathbf{1}_S \mathbf{1}_T' + \mathbf{1}_T \mathbf{1}_S') \\ \sum_{S, T} p_X(S, T) &= 1 \\ \forall S, T \quad p_X(S, T) &\geq 0. \end{aligned}$$

Lifting optimization problems to higher dimensions can make them better-behaved (can help overcome the low-dimensional bottleneck).

The big data paradox in clinical practice
P. Msaouel (2022)

The more data you have, the less likely your confidence intervals are to contain the true value: more data reduces variance (and confidence intervals), but the bias remains. Try to:

- Improve data quality;
- Model subject heterogeneity (missing confounders, e.g., in the linear regressions to analyze RCT results);
- Include the bias in the confidence intervals.

Fine-tuning language models with just forward passes
S. Malladi et al.

Backpropagation of large models requires an inordinate amount of memory (with an 80GB GPU, you can use a

30B model, but only train a 3B one). Use a stochastic gradient estimator (SPSA, ZO-SGD).

$$\begin{aligned} z &\sim N(0, I) \\ \hat{\nabla} \ell(\theta) &= \frac{\ell(\theta + \varepsilon z) - \ell(\theta - \varepsilon z)}{2\varepsilon} z \\ &\approx z z' \nabla_{\theta} \ell(\theta) \end{aligned}$$

To reduce memory further, do not store z , but only its random seed, and re-generate it each time it is needed (MeZO).

Keeping gradient history does not require much additional memory ($\Delta \ell$ and the seed for z for each timestep). For more precision, use several vectors z .

Also try variance reduction, or layerwise gradients (if the layer gradients have different scales).

ChatGPT informed graph neural network for stock movement prediction
Z. Chen

ChatGPT for named entity recognition (NER) and sentiment extraction. (Also add stock embeddings from a co-mention graph, and finish with an LSTM.)

FinGPT: open-source financial large language models
H.B. Yang et al.

LoRA over LLaMA, with daily updated financial text data (news (Yahoo finance), social media (Twitter), SEC filings, blogs, etc.).

Temporal data meets LLM: explainable financial time series forecasting
X. Yu et al.

Ask ChatGPT to:

- Describe the business of a company;
- List its risk, growth, etc. factors;
- Summarize recent (time-stamped) news about it;
- Summarize recent macro-economic news related to the keywords;
- Use all of the above to forecast the future stock price, reasoning step by step.

Bloated disclosures: can ChatGPT help investors process financial information?
A.G. Kim et al.

Ask ChatGPT to summarize corporate disclosures:

- Bloated disclosures (smaller summaries) herald bad market events;
- The sentiment is easier to extract from the summaries.

Improved financial forecasting via quantum machine learning
S. Thakkar et al.

Sampling from a determinantal point process (DPP) (which increases diversity) gives an unbiased least squares estimator

$$\mathbb{E}_{S \sim \text{d-DPP}(XX')} [X_S^{-1} y_S] = \underset{w}{\text{Argmin}} \|Xw - y\|^2$$

Apply this to random forests, sampling both observations and features.

**Quantum computer based
feature selection in machine learning
G. Hellstern et al.**

Let ρ_{ij} be the dependency between predictors i and j , and ρ_{iY} the dependency between predictor i and target Y . The feature selection problem can be formulated as a QUBO problem:

$$\begin{aligned} &\text{Find} \quad z \in \{0, 1\}^n \\ &\text{To maximize} \quad \sum_i z_i |\rho_{iY}| - \lambda \sum_{i \neq j} z_i z_j |\rho_{ij}| \end{aligned}$$

This can be solved with gate-based quantum computers (QAOA).

**A unified framework
for fast large-scale portfolio optimization
W. Deng et al. (2023)**

IPCA (instrumental PCA) is a factor model whose loadings β are linear functions of (time-varying) asset characteristics.

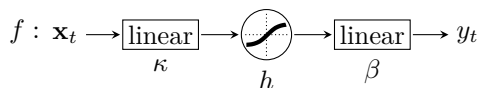
Add ℓ^1 and ℓ^2 penalties to your portfolio optimization problem.

Implementation in PyPortOpt.

**Enhanced Bayesian neural network
for macroeconomics and finance
N. Hauzenberger et al. (2023)**

Nowcast a time series using a Bayesian neural net (separating linear and nonlinear effects, allowing for heteroskedasticity)

$$\begin{aligned} y_t &= \gamma' \mathbf{x}_t + f(\mathbf{x}_t) + \varepsilon_t \\ \varepsilon_t &\sim N(0, \sigma_t^2) \\ \log \sigma_t &\sim \text{AR}(1) \end{aligned}$$



where h is a convex combination of ReLU, LeakyReLU, sigmoid and tanh, with priors

– $\beta \sim \text{MGP}$ (multivariate Gamma process)

$$\begin{aligned} \rho_r &\sim \text{Gamma}(a, 1) \\ \phi_q &= \prod_{1 \leq r \leq q} \rho_r \\ \beta_q &\sim N(0, \phi_q^{-1}) \end{aligned}$$

so that the neurons' activations are progressively shrunk to zero ($\phi_q \gg 1$ for $q \gg 1$): you can have a large number of neurons, only those useful will be kept.

– $\kappa \sim \text{Horseshoe}$

$$\begin{aligned} j &: \text{predictor} \\ q &: \text{neuron} \\ \phi_{jq} &\sim \text{HalfCauchy} \\ \lambda_q &\sim \text{HalfCauchy} \\ \kappa_{jq} &\sim N(0, \lambda_q^2 \phi_{jq}^2) \end{aligned}$$

The Horseshoe prior generalizes the (sparsifying) Laplace prior:

Laplace prior	Student t prior
$\lambda^2 \sim \text{Exp}(2)$	$\lambda^2 \sim \text{IG}(a, b)$
$\beta \sim N(0, \lambda^2)$	$\beta \sim N(0, \lambda^2)$.

– $\beta \sim \text{Horseshoe}$.

**Successive one-sided Hodrick-Prescott filter
with incremental filtering algorithm
for nonlinear economic time series
Y. Liu et al.**

The HP filter is

$$F = \begin{bmatrix} 1 & -2 & 1 & 0 \\ & \diagdown & \diagdown & \diagdown \\ 0 & & 1 & -2 & 1 \end{bmatrix}$$

$$S = I + \eta F' F$$

$$\begin{aligned} \text{trend} &= \underset{u}{\text{Argmin}} \|y - u\|^2 + \eta \|Fu\|^2 \\ &= S^{-1} y \\ \text{cycle} &= (I - S^{-1})y. \end{aligned}$$

It can be computed efficiently on an expanding window (Woodbury formula). It can be iterated, to remove any trend remaining in the cycle:

$$\begin{aligned} \text{cycle} &= (I - S^{-1})^n y \\ \text{trend} &= [I - (I - S^{-1})^n] y \end{aligned}$$

**Generating synergistic formulaic alpha
collections via reinforcement learning
S. Yu et al.**

RL-based formulaic alpha mining to maximize the performance, not of each alpha, but of their combination.

**Efficient solution
of portfolio optimization problems
via dimension reduction and sparsification
C.K. Buhler et al.**

To solve large (Markowitz) portfolio optimization problems approximately and efficiently, try:

– LSTM to forecast if a stock is never (0) or sometimes (1) in the optimal portfolio, for no/some value of the risk aversion λ ;

- Change the risk measure from

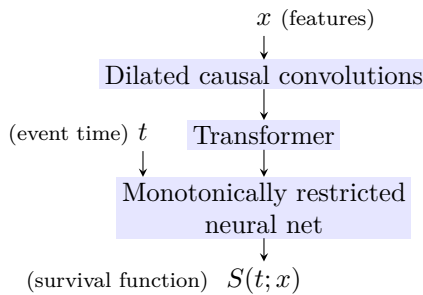
$$w'Vw = w'(X - \bar{X})'(X - \bar{X})w = \|(X - \bar{X})w\|_2^2$$

to $\|(X - \bar{X})w\|_1$, to get a linear program;

- Sparsify the covariance matrix by truncating small values and adding back those in a row or column with a non-zero off-diagonal element (the result is block-diagonal and positive semi-definite).

Deep attentive survival analysis in limit order books: estimating fill probabilities with convolutional-transformers
Á. Arroyo et al.

Use a neural network



for survival analysis, maximizing the right-censored log-likelihood

$$\text{loglik} = \sum_{k \text{ observed}} \log f_{\theta}(t_k; x_k) + \sum_{k \text{ censored}} \log S(t_k; x_k)$$

t_k : event time
 x_k : features

$$S(t) = P[T \geq t] = \exp - \int_0^t h \quad \text{survival function}$$

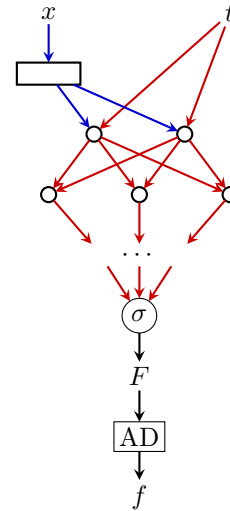
$$F(t) = 1 - S(t) \quad \text{cdf}$$

$$f(t) = -\frac{dS}{dt} \quad \text{pdf}$$

$$h(t) = \frac{f(t)}{1 - F(t)} \quad \text{hazard rate}$$

Neural likelihoods via cumulative distribution functions
P. Chilinski and R. Silva

To model probability distributions (cdf and pdf), use a neural network, with positive weights and tanh activations; automatic differentiation gives f and F . This can be generalized to higher dimensions. Applications include the tail dependence coefficient (TDC).



Agent market orders representation through a contrastive learning approach
R. Ruan et al.

To cluster traders, *i.e.*, identify trading behaviours, consider sequences of 50 market orders by several traders, and use an LSTM with contrastive learning (triplet loss, with positive/negative pairs if the sequences come / do not come from the same trader) to learn a latent representations, and use k -means on it. Plot with t -SNE, or hour \sim date | cluster.

Causal feature engineering of price directions of cryptocurrencies using dynamic Bayesian networks
R. Amirzadeh et al.

DBN (dynamic Bayesian network) to predict the price direction of 5 altcoins from 23 features (technical analysis, gold, MSCI, S&P, USD, oil, number of tweets).

Random matrix theory and nested clustered portfolios on Mexican markets
A. Garcia and B. Rodríguez

Nested clustered optimization (NCO) for $n = 28$:

- Start with the variance matrix (sample, shrinkage, or cleaned with RMT);
- Compute the corresponding minimum spanning tree, its normalized Laplacian L , and the spectrum of L ;
- Only keep the eigenvectors before the maximum spectral gap, $\text{Max}_k |\lambda_k - \lambda_{k-1}|$;
- Apply k -means on the top eigenvectors;
- Compute the optimal portfolio inside each cluster;
- Compute the optimal portfolio of clusters.

Model-free market risk hedging using crowding networks
V. Zlotnikov et al. (2023)

Crowding measures whether active managers are overweight a stock (other measures include: proportion of

buy ratings, institutional trade persistence, momentum, high but achievable expectations). It can be estimated with the centrality in the overweight (resp. underweight) graph built from holdings data.

Causality between sentiment and cryptocurrency prices
L. Mondal et al.

Fit a topic model on crypto-related tweets (a variant of LDA targetted at shorter texts, only allowing one topic per text) and compute topic-level sentiment (BERTweet and Pysentimiento).

Coloring in R's blind spot
A. Zeileis and P. Murrell

R 4.0.0 provides qualitative palettes in `palette.colors` and sequential and diverging ones in `hcl.colors`.

The little book of deep learning
F. Fleuret (2023)

Non-technical (but broad) introduction to deep learning with clear diagrams for the most common architectures (ResNet, UNet, Transformer, etc.).

Descent steps of a relation-aware energy produce heterogeneous graph neural networks
H. Ahn et al. (2022)

One can learn a latent representation Y of the nodes of a graph as

$$Y_w(X) = \underset{Y}{\operatorname{Argmin}} \|Y - f_w(X)\|^2 + \lambda \operatorname{tr}(Y'LY),$$

i.e., nodes with similar features X have a similar representation (first term), and nodes linked by an edge have a similar representation (second term). The weights w of the neural network f_w are chosen wrt some classification or prediction task

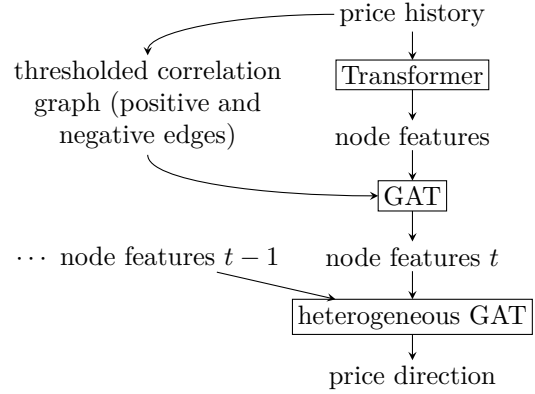
$$\theta, w = \underset{\theta, w}{\operatorname{Argmin}} \sum_i \operatorname{loss}(g_\theta(Y_w(x_i)), y_i).$$

For heterogeneous graphs, consider

$$Y_w(X) = \underset{Y}{\operatorname{Argmin}} \sum_s [\|Y_s - f_{w_s}(X_s)\|^2 + \lambda \sum_{s'} \sum_t \sum_{ij \in E_t} \|Y_i H_t - Y_j\|^2]$$

where s is the node type, t the edge type, H_t compatibility matrices (to allow heterophily).

Temporal and heterogeneous graph neural network for financial time series prediction
S. Xiang et al. (2022)



Limitations of deep learning for inverse problems on digital hardware
H. Boche et al.

There are only countably many computable real numbers (because there are only countably many Turing machines).

There is no computable mapping

$$(A, y) \mapsto \underset{x \in \mathbb{C}^N}{\operatorname{Argmin}} \|x\|_1 \text{ such that } \|Ax - y\|_2 \leq \varepsilon.$$

There are algorithms to solve that problem, but they come without explicit error bounds: they do not have an effective stopping criterion.

The result generalizes to oracle Turing machines: Turing machines dealing with arbitrary real numbers via rational Cauchy sequences provided by an oracle.

Inverse problems are solvable on real number signal processing hardware
H. Boche et al.

Blum-Shub-Smale (BSS) machines (Turing machine operating on real numbers) are more powerful than oracle Turing machines.

All you need is good init
D. Mishkin and J. Matas

LSUV (layer sequential unit variance) first initializes weights with orthogonal matrices, and then adjusts them (via optimization) to ensure the activations have unit variance.

Fixup initialization: residual learning without normalization
H. Zhang et al.

Resnet output variance grows exponentially with depth: fixup (fixed update) initialization

- Initializes the layers with a classical method;
- Rescales the layers in the residual branches (those in the shortest path are left untouched);
- Sets the last layer of each residual branch to 0;

- Adds learnable scalar multiplier (initialized at 1) and bias (0) before each layer (convolution, linear, elementwise nonlinearity).

Improving transformer optimization through better initialization
X.S. Huang et al. (2020)

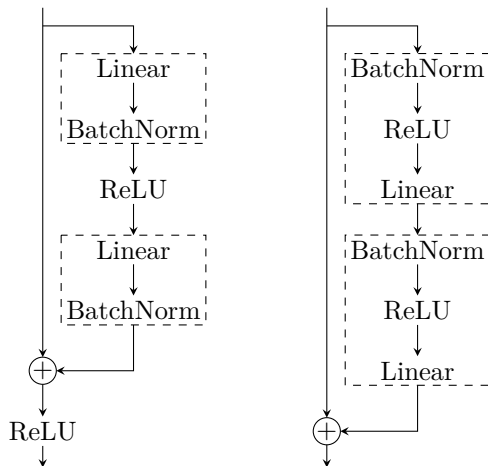
With T-Fixup initialization, deep transformer networks no longer require learning rate warmup and LayerNorm.

How does batch normalization help optimization
S.Santurkar et al. (2018)

The success of BatchNorm is not so much due to the reduction of the “internal covariate shift” but to the smoothing of the loss landscape.

Identity mappings in deep residual networks
K. He et al. (2016)

Change the order of the layers in ResNet.



TrivialAugment: tuning-free yet state-of-the-art data augmentation
S.G. Müller and F. Hutter

For each image, pick one (instead of several) augmentation, uniformly at random (instead of from a learned distribution), and apply it with a random (instead of learned) strength.

A study on the evaluation of generative models
E. Betze et al.

Do not use the inception score and ImageNet to assess generative models:

- Use NotImageNet32 instead of ImageNet;
- Use CLIP instead of the inception model
- Use several metrics: FID_{∞} (unbiased FID), KID, ClearFID.

Elucidating the design space of diffusion-based generative models
T. Karras et al. (2022)

A denoising diffusion model can be written as an ODE

$$\dot{x} = -\dot{\sigma}_t \sigma_t \nabla_x \log p(x, \sigma_t)$$

(we have many trajectories $(x_t)_t$ from which we want to learn the score function $\nabla \log p$). To sample, use higher-order methods (Euler-Heun, Runge-Kutta). Stochastic sampling (Langevin diffusion) gives better results.

MaskGIT: masked generative image transformer
H. Chang et al.

Vision transformers (ViT) generate image patches sequentially, in raster scan order. Instead, learn to predict randomly masked patches (à la BERT), generate the image all at once, and progressively refine it.

On the importance of noise scheduling for diffusion models
T. Chen

The optimal noise schedule depends on the image size: there is more redundancy in large images.

Texture generation with neural cellular automata
A. Mordvintsev et al.

Train a neural cellular automaton (NCA) (all cells have the same differentiable update rule) to produce a (tilable) pattern given an example, using a style transfer loss (Gram matrix of the latent representation of a classifier) or the inception distance.

Ensemble deep learning: a review
M.A. Ganaie

Ensemble learning is not limited to equal weighted averages of a few models:

- Bagging;
- Boosting;
- Stacking;
- Negative correlation learning, *i.e.*, adding a penalty to make the base learners more diverse (they should not all make the same errors);
- Implicit ensembles (dropout, drop-connect, stochastic depth)
- Homogeneous ensembles (same models, but different training datasets, augmentations, initializations, regularizations, hyperparameters), heterogeneous ensembles;
- Weighted average, with learned weights (mixture of experts), or weights chosen by cross-validation (SuperLearner)
- Voting rules (Condorcet, etc.)

**Loss surfaces, mode connectivity,
and fast ensembling of DNNs**
T. Garipov et al. (2018)

The optima of loss functions are connected by simple curves over which training and test accuracy are nearly constant: learn those curves (e.g., Bézier) ϕ_θ by minimizing

$$\mathbb{E}_{t \sim U(0,1)} [\text{loss}(\phi_\theta(t))].$$

Fast geometric ensembling (FGE) starts with a single solution and looks for a nearby solution (sufficiently away from it) with a similar loss – this can be done using a sawtooth learning rate schedule.

**A convergent ADMM framework
for efficient neural network training**
J. Wang et al. (2015)

The neural network training problem can be written in ADMM form

$$\begin{array}{lll} \text{Find} & w_1, \dots, w_L & \text{weights} \\ & a_1, \dots, a_L & \text{activations} \\ \text{To minimize} & \text{loss}(a_L, y) + \sum_\ell r_\ell(w_\ell) & \\ \text{Such that} & \forall \ell \in \llbracket 1, L \rrbracket \quad a_\ell = g_\ell(w_\ell, a_{\ell-1}) & \end{array}$$

or, by replacing the non-linear constraints with a penalty, minimizing

$$\text{loss}(a_L, y) + \sum_\ell r_\ell(w_\ell) + \beta \sum_\ell \|a_\ell - g_\ell(w_\ell, a_{\ell-1})\|_2^2.$$

Update the parameters first backwards

$$a_L \rightarrow w_L \rightarrow a_{L-1} \rightarrow w_{L-1} \rightarrow \dots \rightarrow a_1 \rightarrow w_1$$

then forwards

$$w_1 \rightarrow a_1 \rightarrow w_2 \rightarrow a_2 \rightarrow \dots \rightarrow w_L \rightarrow a_L.$$

**Fast and flexible ADMM algorithms
for trend filtering**
A. Ramdas and R.J. Tibshirani

Trend filtering solves the problem

$$\text{Minimize}_\beta \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|D^{(k+1)}\beta\|_1.$$

The standard ADMM approach rewrites the problem as

$$\begin{array}{lll} \text{Find} & \beta, \alpha & \\ \text{To minimize} & \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|\alpha\|_1 & \\ \text{Such that} & \alpha = D^{(k+1)}\beta. & \end{array}$$

Instead, rewrite it as

$$\begin{array}{lll} \text{Find} & \beta, \alpha & \\ \text{To minimize} & \frac{1}{2} \|y - \beta\|_2^2 + \lambda \|D^{(1)}\alpha\|_1 & \\ \text{Such that} & \alpha = D^{(k)}\beta. & \end{array}$$

The 0th order trend filtering problem, $k+1=1$, can be solved in linear time, with dynamic programming, or the taut string principle.

**The Gromov-Wasserstein distance between
networks and stable network invariants**
S. Chowdhury and F. Mémoli

Given two metric measure spaces (X, d_X, p_X) , (Y, d_Y, p_Y) , their Gromov-Wasserstein distance is the minimum distortion of a coupling between p_X and p_Y .

$$\inf_{\substack{\pi_1 \# p_X = p_Y \\ \pi_2 \# p_Y = p_X}} \iint_{X \times Y \times X \times Y} |d_X(x, x') - d_Y(y, y')|^q p(dx, dy) p(dx', dy')$$

It can be generalized to *networks* (X, ω_X, p_X) , where X is a Polish space, ω_X a measurable function on $X \times X$ (it need not satisfy the triangle inequality, be symmetric, or have zeros on the diagonal).

Weisfeiler-Lehman meets Gromov-Wasserstein
S. Chen et al.

A *labeled measure Markov chain* (LMMC) is the datum of:

- X , a finite set (states);
- $m : X \rightarrow \mathcal{P}(X)$ (Markov kernel, *i.e.*, transition matrix, where $\mathcal{P}(X)$ is the set of probability distributions on X);
- $\mu \in \mathcal{P}(X)$ a stationary distribution for m ;
- $\ell : X \rightarrow Z$ (label, for some metric space Z).

A labeled graph, and $q \in [0, 1]$, defines an LMMC on the set of nodes, with transition probabilities

$$m(u, v) = \begin{cases} 1 & \text{if } u = v \text{ and } \mathcal{N}_u = \emptyset \\ q & \text{if } u = v \text{ and } \mathcal{N}_u \neq \emptyset \\ \frac{1-q}{\deg u} & \text{if } v \in \mathcal{N}_u \\ 0 & \text{otherwise} \end{cases}$$

and stationary distribution $\mu(v) \propto \text{Max}\{1, \deg v\}$.

The multiset labels in the WL test can be interpreted as probability distributions $\mathcal{L}^1, \mathcal{L}^2, \dots, \mathcal{L}^k$ on $\mathcal{P}(Z)$, $\mathcal{P}\mathcal{P}(Z), \dots, \mathcal{P} \dots \mathcal{P}(Z) = \mathcal{P}^{\circ k}(Z)$. The WL distance between X and Y is

$$d^k(X, Y) = d_W(\mathcal{L}_X^k, \mathcal{L}_Y^k)$$

(Wasserstein distance on $\mathcal{P}^{\circ k}(Z)$).

The open algebraic path problem
J. Master

The shortest path problem on a graph computes the transitive closure of the weight matrix with coefficients in the *rig* $([0, \infty], \text{Min}, +)$. The algebraic shortest path generalizes this to an arbitrary *rig* or even a commutative quantale (monoidal closed poset with all joins).

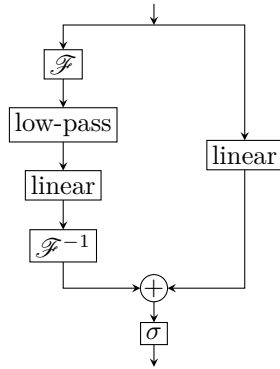
$$\begin{array}{llll} [0, \infty] & \geq & \inf & + \\ [0, \infty] & \leq & \sup & \inf \\ [0, 1] & \leq & \sup & \times \\ \{\mathbf{T}, \mathbf{F}\} & & \text{OR} & \text{AND} \\ \mathcal{P}(\Sigma^*) & \subset & \cup & \text{CONCAT} \end{array}$$

Uncertainty in deep learning
Y. Gal (2016)

To get a probabilistic forecast (as opposed to a point forecast) from a neural net trained with dropout (or some other stochastic regularization technique), use the network at test time as you did during training, to sample from the distribution of y^* and estimate $E[y^*]$ and $\text{Var}[y^*]$.

**Fourier neural operator
for parametric partial differential equations**
Z. Li et al. (2021)

Neural PDE solver, in Fourier space: repeat 4 times, after increasing the dimension with a linear layer.



Clifford neural layers for PDE modeling
J. Brandsetter et al. (2023)

To solve PDEs involving “correlated fields” (e.g., a scalar field and a vector field: temperature and wind speed), model them jointly, as a multivector, and use Clifford convolutions and Clifford layers.

**Deep learning for solving and estimating
dynamic macro-finance models**
B. Fan et al.

To solve a PDE

$$\begin{cases} f_\theta(x, u, Du) = 0 & x \in \Omega \\ B(x, u) = 0 & x \in \partial\Omega \end{cases}$$

train a neural net to minimize

$$\sum_{x \in T_f} \|f_\theta\|^2 + \sum_{z \in T_b} \|B\|^2$$

(PINN: physically-informed neural net). One can add other conditions

$$I(u, x) = 0 \quad c \in I \subset \Omega$$

and learn the PDE parameters θ as well.

Application to dynamic equilibrium models of the financial sector. Implementation with DeepXDE.

Nowcasting with signature methods
S.N. Cohen et al. (2023)

The Kalman filter can be expressed as a regression on signature transform features (involving only linear features).

Use the signature transform features ψ (on a moving window), which allow for irregular sampling, for nowcasting.

$$y_t = \sum_k (\alpha_k + \beta_k y_{t-}) \psi_{k,t} + \varepsilon_t$$

where y_{t-} is the previous value. Add an elasticnet penalty, and standardize the features (otherwise, they decrease factorially fast).

Implementation in SigNow; also check esig, iisignature, signatory.

**A new approach to decomposition of economic
time series into permanent and transitory
components with particular attention to
measurement of the ‘business cycle’**
S. Beveridge and C.R. Nelson

A stochastic process X , with stationary $\text{MA}(\infty)$ increments $\Delta X = \psi(L)\varepsilon$, can be decomposed into a stochastic trend (a random walk: its increments are independent)

$$\begin{aligned} \text{Trend}_t &= \lim_{s \rightarrow \infty} E[X_{t+1} | \mathcal{F}_t] \\ &= \psi(1) \sum_{s \leq t} \varepsilon_s \end{aligned}$$

and a cycle component.

$$\begin{aligned} (1 - L)X &= \psi(L)\varepsilon \\ &= \psi(1)\varepsilon + [\psi(L) - \psi(1)]\varepsilon \end{aligned}$$

**A unified approach for jointly estimating
the business and financial cycle
and the role of financial factors**
T. Berger et al. (2022)

The Beveridge-Nelson decomposition of a $\text{VAR}(1)$ process $\Delta X_t = A\Delta X_{t-1} + \varepsilon_t$ is

$$\begin{aligned} \text{Trend}_y &= \lim_{s \rightarrow \infty} E[X_{t+1} | \mathcal{F}_t] \\ &= X_t + A(I - A)^{-1} \Delta X_t. \end{aligned}$$

Trend-cycle decompositions
E. Zivot (2005)

In the $\text{MA}(\infty)$ representation of an $\text{ARIMA}(p, 1, q)$ process

$$\begin{aligned} \phi(L)\Delta X &= \theta(L)\varepsilon & \text{ARMA} \\ \Delta X &= \psi(L)\varepsilon & \text{MA}(\infty) \end{aligned}$$

$\psi(1)$ is the permanent effect of a shock ε_t on the level of X_t .

The Beveridge-Nelson trend is

$$\text{Trend}_t = \text{Trend}_0 + \psi(1) \sum_{s=1}^t \varepsilon_s.$$

The unobserved component model (UC-ARMA) is

$$\begin{aligned} X_t &= \text{Trend}_t + \text{Cycle}_t \\ \text{Trend}_t &= \mu + \text{Trend}_{t-1} + \varepsilon_t \quad \varepsilon_t \sim N(0, \Sigma) \\ \phi(L)\text{Cycle} &= \theta(L)\eta \quad \eta_t \sim N(0, S). \end{aligned}$$

If the cycle is a stationary AR process, *i.e.*, $\theta(L) = 1$, and the roots of ϕ are outside the unit circle (and $\varepsilon \perp \eta$), it can be written as a state space model

$$\begin{bmatrix} c_t \\ \vdots \\ c_{t-p+1} \end{bmatrix} = \begin{bmatrix} \phi_1 & \dots & \phi_p \\ 1 & 0 & \dots & 0 \\ 0 & \diagdown & & \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} c_{t-1} \\ \vdots \\ c_{t-p} \end{bmatrix} + \begin{bmatrix} \eta_t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\Delta X_t = \mu - \mathbf{1}' \begin{bmatrix} c_{t-1} \\ \vdots \\ c_{t-p} \end{bmatrix} + \varepsilon_t$$

and estimated with a Kalman filter.

The *variance ratio* statistic

$$\begin{aligned} R_k &= V_k / V_1 \\ V_k &= \frac{1}{k} \text{Var}[X_{t+k} - X_t - k\mu] \\ \mu &= E[\Delta X_t] \end{aligned}$$

measures the fraction of variation due to permanent shocks.

Bayesian vector autoregressions

T. Woźniak

A VAR(p) model

$$t_t = \mu + A_1 y_{t-1} + \dots + A_k y_{t-k} + u_t \quad u_t \sim N(0, \Sigma)$$

can be written, by transposing the equation and vertically stacking the resulting row vectors,

$$Y = XA + U \quad U \sim \text{MN}(0, \Sigma, I).$$

With a normal-Wishart prior

$$\begin{aligned} \Sigma &\sim \text{IW}(\underline{S}, \underline{\nu}) \\ A|\Sigma &\sim \text{MN}(\underline{A}, \Sigma, \underline{V}), \end{aligned}$$

the posterior is also normal-Wishart

$$\begin{aligned} \Sigma &\sim \text{IW}(\overline{S}, \overline{\nu}) \\ A|\Sigma &\sim \text{MN}(\overline{A}, \Sigma, \overline{V}), \end{aligned}$$

where

$$\begin{aligned} \overline{V} &= (\underline{V}^{-1} + X'X)^{-1} \\ \overline{A} &= \overline{V}(\underline{V}^{-1}\underline{A} + X'Y) \\ \overline{\nu} &= \underline{\nu} + T \\ \overline{S} &= \underline{S} + Y'Y + \underline{A}'\underline{V}^{-1}\underline{A} - \overline{A}'\overline{V}^{-1}\overline{A}. \end{aligned}$$

The Minnesota prior is

$$A = \begin{bmatrix} 0 & \dots & 0 \\ 1 & & & \\ & \diagdown & & \\ & & 1 & \\ 0 & & & \diagdown & \\ & & & & 0 \\ \vdots & & & & \vdots \\ 0 & & & & \diagdown & \\ & & & & & 0 \end{bmatrix}$$

$$\underline{V} = \text{diag}\left(\lambda_0, \frac{\lambda_1}{1^2 \hat{\sigma}_1^2}, \dots, \frac{\lambda_1}{1^2 \hat{\sigma}_n^2}, \frac{\lambda_1}{2^2 \hat{\sigma}_2^2}, \dots\right)$$

where $\hat{\sigma}_i^2$ is the error term variance of a univariate AR(k) model on variable i . For the other hyperparameters, try

$$\begin{aligned} \underline{S} &= (Y - XA)'(Y - XA) \\ \underline{\nu} &= T - K - N - 1. \end{aligned}$$

Nowcasting with large Bayesian vector autoregressions J. Cimadomo (2020)

A *state space BVAR* is a VAR written in state space form. To sample from it, iterate:

- Sample the missing values given A and Σ , using a Kalman simulation smoother;
- Sample the Minnesota hyperparameters given the data;
- Sample the parameters A , Σ given the data and the hyperparameters.

A *stacking BVAR* deals with mixed frequency data by adding the lags of the high-frequency variables.

The *cube root BVAR* deals with monthly and quarterly data by fitting a VAR(p) on quarterly data, writing it in companion form, $Y_t = AY_{t-1} + \varepsilon_t$ and converting it to a monthly VAR, $Y_t = BY_{t-1} + \eta_t$, where $B^3 = A$. There are many cubic roots of A : for the real eigenvalues, take the real cubic root; for the complex one, take that with the lowest argument (*i.e.*, that leading to the least oscillatory behaviour).

Large Bayesian vector autoregressions M. Bańbura et al. (2007)

The FAVAR model is a VAR model with a small number of variables, augmented with a few principal components from a larger number of variables.

Estimating and accounting for the output gap with large Bayesian vector autoregressions
J. Morley and B. Wong (2019)

To choose the variables to include in a BVAR (reduce their number from 138 to 7), look at the contribution of each to the Beveridge-Nelson cycle (adding (removing) irrelevant (relevant) variables does not (does) change the cycle) and remove the variables one by one, starting with those whose contribution has the lowest variance.

Use a Minnesota prior

$$\begin{aligned} E[\phi_k^{ik}] &= 0 \\ \text{Var}[\phi_k^{ij}] &= \frac{\lambda^2}{k^2} \text{ if } i = j \\ \text{Var}[\phi_k^{ij}] &= \frac{\lambda^2}{k^2} \frac{\sigma_i^2}{\sigma_j^2} \text{ if } i \neq j \end{aligned}$$

where λ is selected to minimize the 1-step ahead RMSE on an expanding window.

The contributions can be computed from

$$\begin{aligned} (\Delta X_t - \mu) &= F(\Delta X_{t-1} - \mu) + He_t \\ \text{Cycle}_t &= -F(I - F)^{-1}(\Delta X_t - \mu) \\ &= \dots \\ &= -\sum_{I \geq 0} F^{I+1}(I - F)^{-1}He_{t-I}. \end{aligned}$$

Vector autoregressions: forecasting and reality
J.C. Robertson and E.W. Tallman (1999)

The VAR(1) model (e.g., a BVAR with a Minnesota prior) $X_t = AX_{t-1} + \varepsilon_t$, $\varepsilon_t \sim N(0, \Sigma)$ can be written $X_t \sim N(AX_{t-1}, \Sigma)$. If some of the coordinates are not observed, their distribution given the observed coordinates is known (*conditional forecasting*) (from the usual formula for the conditional Gaussian distribution).

$$\begin{aligned} E[X_t^{\text{mis}} | X_t^{\text{obs}}] &= \\ &= (AX_{t-1})^{\text{mis}} + \Sigma_{\text{mis,obs}} \Sigma_{\text{obs,obs}}^{-1} [X_t^{\text{obs}} - (AX_t)^{\text{obs}}] \end{aligned}$$

Large Bayesian vector autoregression
J.C.C. Chan

The BVAR with errors $U \sim \text{MN}(0, \Sigma, \Omega)$ can be non-Gaussian and/or have autocorrelated errors:

- Gaussian: $\Omega = I$;
- Common stochastic volatility:

$$\begin{aligned} \Omega &= \text{diag}(e^{h_1}, \dots, e^{h_T}) \\ h_t &= \rho h_{t-1} + \varepsilon_t \end{aligned}$$

- Fat tails (Student T):

$$\begin{aligned} \Omega &= \text{diag}(\lambda_1, \dots, \lambda_T) \\ \lambda_t &\sim \text{InverseGamma} \end{aligned}$$

- Serially dependent errors, e.g., $u_t \sim \text{MA}(2)$, i.e.,

$$\Omega = \begin{pmatrix} & & & 0 \\ & \diagdown & \diagdown & \diagdown \\ & \diagdown & \diagdown & \diagdown \\ 0 & \diagdown & \diagdown & \diagdown \end{pmatrix}$$

Industry classification using a novel financial time series case representation
R. Dolphin et al.

Compute an embedding of stocks from pricing data:

- S_{ij} = similarity between the delay embeddings ($\tau = 5$) of the daily return time series
- C_{ij} = number of times stock i is among the k nearest neighbours of stock j ($k = 1$);
- $C \approx EE'$, low-rank factorization, ignoring the diagonal (after log-transforming the entries of C).

Visualize the truncated graph with t-SNE.

Hedonic prices and quality-adjusted price indices powered by AI
P. Bajari et al. (2023)

Forecast product prices from (time-stamped) unstructured product data (text, image: BERT and ResNet50) and use this model to measure *inflation*.

$$\begin{aligned} \text{Price}_{it} &= h_t(\text{features}_{it}) + \varepsilon_{it} \\ \text{or } \text{Price}_{i,\cdot} &= h(\text{features}_i) + \varepsilon_{i,\cdot} \end{aligned}$$

Precision versus shrinkage: a comparative analysis of covariance estimation methods for portfolio allocation
S. Dutta and J. Jain (2023)

To estimate the variance matrix, for portfolio optimization, prefer Gaussian graphical models:

- Graphical lasso

$$\text{Minimize}_{\Theta \succ 0} \log \det \Theta - \text{tr}(\hat{\Sigma} \Theta) - \lambda \|\Theta\|_1$$

- Nodewise regression;
- Clime

$$\text{Minimize}_{\Theta \succ 0} \|\Theta\|_1 \text{ such that } \left\| \hat{\Sigma} \Theta - I \right\|_{\infty} \leq \lambda$$

- Greedy prune: find the neighbours of i by adding the variable X_j minimizing $\text{Var}[X_1 | X_{\mathcal{N}(i)}, X_j]$, T times, and then pruning those contributing to a proportion less than ν to the variance;
- Hybrid MB: nodewise regression with no L^1 penalty on the nearest neighbour, followed by pruning.

Alternatives include

- Linear shrinkage (for various choices of the target and the shrinkage constant: Ledoit-Wolf, Rao-Blackwell-Ledoit-Wolf, Bodnar, oracle);

- Non-linear shrinkage (shrink the eigenvalues of $\hat{\Sigma}$ separately, to minimize some loss function);
- Thresholding (of the covariances): hard, soft, or adaptive;
- Random matrix theory (rotationally invariant estimators).

Non-parametric cumulants approach for outlier detection of multivariate financial data
F. Cesarone et al. (2023)

The CGF-PCA (cumulant generating function) direction is the solution θ of

$$\underset{\theta : \|\theta\|=1}{\text{Maximize}} \log E[e^{r\theta'X}]$$

for a fixed $r > 0$. We are maximizing (not minimizing) a convex function on the sphere (a non-convex set) – but we can replace $\|\theta\| = 1$ with $\|\theta\| \leq 1$ and solve with multistart and projected gradient.

Identify outliers by looking at the projection onto $\text{Span } \theta$.

Multivariate range value-at-risk and covariance risk measures for elliptical and log-elliptical distributions
B. Zuo et al.

The multivariate range VaR is

$$E[X \mid \forall i \text{ VaR}_{p_i} X_i \leq X_i \leq \text{VaR}_{q_i} X_i].$$

The multivariate range covariance is

$$E[(X - \text{MRVaR}_{pq})(X - \text{MRVaR}_{pq})' \mid \text{VaR}_p X \leq X \leq \text{VaR}_q X].$$

Market making with deep reinforcement learning from limit order books
H. Guo et al.

The state space contains the current LOB (limit order book: price, volume, ask, bid, several levels), order imbalance (several windows), realized volatility, technical indicators, inventory, remaining time. Try to forecast the price direction $[t - k, t]$ vs $[t, t + k]$. The reward includes a damped PnL

$$(1 - \lambda)(\Delta \text{PnL})_+ - (\Delta \text{PnL})_-$$

and an inventory penalty. Train with Dueling DQN or PPO.

Trending fast and slow
E. Cheng et al.

Trend following strategies with a volatility dependent lookback (1m vs 12m): slow momentum in low volatility and fast momentum in high volatility work better.

LLaMa: open and efficient foundation language models
H. Touvron et al. (2023)


LLaMa is a family of large language models (LLM), from 13B to 5B parameters, designed to be inexpensive at inference time, trained using only publically available data:

- CommonCrawl/C4, preprocessed in different ways (deduplication, quality filtering, identification of Wikipedia references);
- Github (BigQuery, filtered for free licences, deduplicated, with boilerplate removal);
- Wikipedia (20 languages, boilerplate removed);
- Arxiv (L^AT_EX files, starting at the first section and ending before the references, with macros removed);
- StackExchange (without HTML, with answers sorted by score);
- Gutenberg and Books3 (deduplicated);

using

- Byte pair encoding (BPE, SentencePiece);
- Prenormalization (normalize the input of the transformers, instead of the output, as GPT3);
- SwiGLU (as PaLM);

$$\text{GLU}(a, b) = a \odot \sigma(b)$$

$$\text{Swish}_\beta(x) = x \odot \sigma(\beta x) = \text{} \quad (\beta \text{ learnable})$$

$$\text{SwiGLU}(x) = \text{Swish}_\beta(xW + b) \odot (xV + c)$$

- Rotary positional embedding (RoPE);
- AdamW (decoupled weight decay regularization), with a cosine learning rate schedule, weight decay, gradient clipping, warmup;

$$\text{Weight decay (Adam)} \quad g = \nabla f(\theta) + w\theta$$

$$\text{Decoupled weight decay} \quad \theta \leftarrow \theta - \eta \left(\frac{m}{\sqrt{v} + \varepsilon} + w\theta \right)$$

- xformers (efficient implementation of the causal transformer);
- 20488 80GB GPUs, over 21 days.

Briefly fine-tuning on instructions data leads to rapid improvements.

Self-instruct: aligning language model with self-generated instructions
Y. Wang et al.

To fine-tune a large language model (LLM) on instruction data, you do not need a lot of data:

- Start with a small seed set of manually written instructions (175 instruction-input-output triplets);
- Ask the model itself to generate new instructions; prune low-quality or duplicated ones;
- Fine-tune the model using those instructions.

Alpaca applies self-instruct to llama.

Training compute-optimal large language models
J. Hoffmann et al. (2022)

For compute-optimal training, model size and training tokens should be scaled equally.

$$\begin{aligned}\#\text{parameters} &\propto \text{FLOP}^{1/2} \\ \#\text{tokens} &\propto \text{FLOP}^{1/2} \\ \#\text{tokens}/\#\text{parameters} &= 20\end{aligned}$$

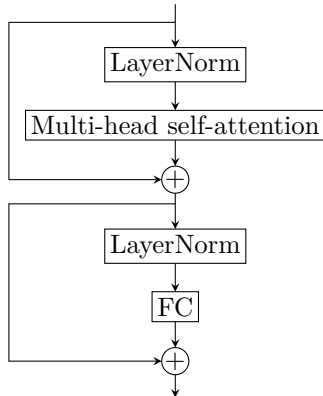
Scaling laws for neural language models
J. Kaplan et al. (2020)

Larger language models make more effective use of training data; the number of parameters should increase faster than the number of tokens.

$$\begin{aligned}\#\text{tokens} &\propto \text{FLOP}^{1/4} \\ \#\text{parameters} &\propto \text{FLOP}^{1/3}\end{aligned}$$

BloombergGPT: a large language model for finance
S. Wu et al. (2023)

Bloom-like model, trained on 50% public (non-financial) data, and 50% Bloomberg data.



- 50B parameters, for 700B tokens, in line with the Chinchilla scaling laws

$$\begin{aligned}\#\text{parameters} &= \text{FLOP}^{1/2}/10 \\ \#\text{tokens} &= \text{FLOP}^{1/2}\end{aligned}$$

- L MSA layers with hidden dimension $D = e^{5+0.06L}$;
- GELU activations, $x \mapsto x\Phi(x)$;
- ALiBi positional encodings;
- Embedding layer normalization;
- AdamW
- Model sharding over 128 GPUs (ZeRO);
- Hierarchical communication and 2-hop gradient updates to reduce communication (MiCS);
- Checkpointing (recompute the activations in the backward pass to reduce memory consumption);
- Mixed precision: FP32 and BF16 (FP32 with truncated mantissa – FP16 truncates both mantissa and exponent);
- Tweak batch size, learning rate, dropout as needed.

BLOOM: a 176B-parameter openaccess multilingual language model
T. Le Scao et al.

Publically available model, trained on 500 multilingual Huggingface datasets (ROOTS corpus), with:

- ALiBi positional encoding (instead of learned or rotary embeddings);
- LayerNorm after the embeddings;
- bfloat16 instead of float16 (which is responsible for some LLM training instabilities);
- Learned multilingual subword tokenizer (BPE) with 250k tokens;
- Distributed learning.

Several models are available, finetuned (BLOOMZ) or not (BLOOM), with 560M, 1.1B, 1.7B, 3B, 7.1B, 176B parameters.

Pythia: a suite for analyzing large language models across training and scaling
S. Biderman et al.

Family of LLMs, with the same structure as GPT-Neo or OPT, from 70m to 12b parameters, each with 154 checkpoints, to study scaling laws.

LoRa: low-rank adaptation of large language models
E. Hu et al.

To efficiently fine-tune large models, replace big weight matrices W with $W + B'A$, freezing W and only fine-tuning the low-rank matrix $B'A$ (initialize them as $A \sim N(0, \sigma^2)$, $B = 0$).

OPT: open pre-trained transformer language models
S. Zhang et al.

GPT3-like model, with 175B parameters (only smaller variants are available, 125B to 66B).

OPT-IML: scaling language model instruction meta learning through the lens of generalization
S. Iyer et al.

Instruction fine-tuning of OPT on 2000 NLP tasks (30B and 175B parameters).

GLM: general language model pretraining with autoregressive blank infilling
Z. Du et al.

LLMs tend to fall in one of three categories: encoder-only (BERT), encoder-decoder (T5), autoregressive (GPT). They can be combined:

- Start with a sentence, $x_1x_2x_3x_4x_5x_6$;
- Mask a few spans (hide the sizes): $x_1x_2[M]x_3[M]$;
- Ask the model to recover them, in an autoregressive way $x_1x_2[M]x_4[M][S]x_5x_6[S]x_3$.

Models available for English and Chinese, with a restrictive license (do not offend the Chinese government).

***Causal reasoning and large language models:
opening a new frontier for causality***
E. Kıcıman et al. (2023)

Causal inference addresses different questions (existence and direction of a causal relation; strength of that relation; full causal graph; attribution: sufficient cause, necessary cause, social norm violation; counterfactual reasoning). On several (small) benchmarks, LLMs perform those tasks better than data-based approaches: *knowledge-based causal inference* answers those questions, by looking, not at the data, but at the metadata (the column names).

To check if the model has seen and memorized the benchmark dataset, provide it with half the column names, or half the questions, and ask for the other half.

To measure the importance of each word in the prompt, remove a few words, and check if the output changes.

***A comprehensive survey
on pretrained foundation models:
a history from BERT to ChatGPT***
C. Zhou et al.

List of models for text, images, or graphs.

***Community detection via semi-synchronous
label propagation algorithms***
G. Cordasco and L. gargano

Label propagation (LPA) computes communities on a graph, by starting to put each node in its own community and then assigning each node to the majority community of its neighbours (breaking ties at random), until convergence. The synchronous version of the algorithm can exhibit cyclic oscillations; the asynchronous version can wrongly produce a giant community.

For bipartite graphs, alternately updating all the nodes of one of the partitions is more stable. This semi-asynchronous algorithm can be generalized to arbitrary graphs, by choosing a node colouring such that adjacent nodes have different colours.

Peirce's tutorial on existential graphs
J.F. Sowa

In MS514, Peirce presents a graphical language for first order logic (FOL).

A *line of identity* represents an existential classifier.

— $\exists x$







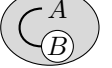
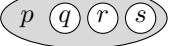
It can be connected to one or several predicates.

— A	$\exists x A(x)$
A — B	$\exists x A(x) \wedge B(x)$
A $\begin{array}{c} \diagup \quad \diagdown \\ B \\ \\ C \end{array}$	$\exists x A(x) \wedge B(x) \wedge C(x)$
S — V — O	$\exists x, y S(x) \wedge O(y) \wedge V(x, y)$



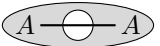

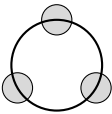
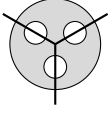
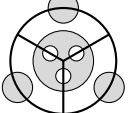
Conjunction is implicit.

— A — B $\exists x \exists y A(x) \wedge B(y)$

Shaded ellipses represent negations.

	$\neg \exists x A(x)$
	$\exists x \neg A(x)$
	$\forall x A(x)$
	$\forall x A(x) \Rightarrow B(x)$
p q	$p \wedge q$
	$p \vee q$
	$p \Rightarrow q$
	$\forall x A(x) \Rightarrow B(x), i.e., A \subset B$
	$p \Rightarrow (q \wedge r \wedge s)$

Lines can be replaced with explicit equalities

—	$\exists x$
— is —	$\exists x \exists y x = y$
— is —	$\exists x \exists y x \neq y$
—  —	$\exists x \exists y x \neq y$
A —  — A	$\exists x \exists y x \neq y \wedge A(x) \wedge A(y)$
	at most one A
	exactly one A
	at least 3
	at most 3
	exactly 3

There are 3 rules of inference:

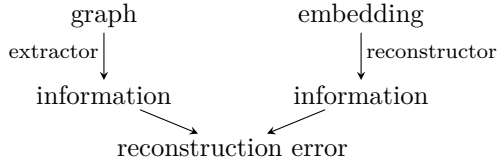
1. You can erase anything in the unshaded areas (e.g., cut lines); you can add anything in the shaded areas

2. Lines can be extended into any enclosed area; lines can be retracted from any enclosed areas; anything can be copied inwards; anything can be deleted outwards;
3. You can add or remove vacant ring-shaped areas (double negations).

Deep learning on graphs

Y. Ma and J. Tang

4. Graph embedding models look for (graph or node) representations preserving some information about the graph or node,



e.g., random-walk-based co-occurrence (DeepWalk is word2vec on random walks) or structural similarity (struc2vec)

$$g_0(v_1, v_2) = 0$$

$$g_k(v_1, v_2) = g_{k-1}(v_1, v_2) + d(sN_k(v_1), sN_k(v_2))$$

where

v_1, v_2 are nodes

$N_k(v)$ is the set of k -hop neighbours of v

$s(N)$ is the degree distribution of the nodes in N

d is the DTW distance

or neighbourhood similarity

$$s_{ij} = \frac{A_i A'_j}{\|A_i\| \|A_j\|}$$

or community membership, or the adjacency matrix

$$A_{ij} \sim \text{Bernoulli}(\sigma(u'_i u_j)).$$

5. GNNs combine filtering layers (which change the features but not the graph structure) and pooling layers (which coarsen the graph).

Filtering layers include

- Spectral filters $f \mapsto U\gamma(\Lambda)U'f$, where

$$L = D - A \text{ or } D^{-1/2}(D - A)D^{-1/2}$$

$$L = U\Lambda U'$$

$$\gamma(\lambda) = (1 + c\lambda)^{-1} \text{ (low pass filter)}$$

or γ is a (learned) Chebychev polynomial;

- Aggregation of the neighbouring values (mean, sum, max, or even LSTM for some (possibly arbitrary) order of the neighbours);
- Attention, *i.e.*, weighted average of the neighbouring values, with data-dependent weights;

- Mixture models (MoNet) are a type of attention where the weights are computed from the degrees

$$c_{ij} = \begin{bmatrix} \sqrt{d_i} \\ \sqrt{d_j} \end{bmatrix}$$

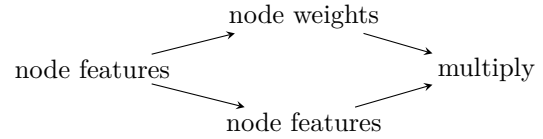
$$\alpha_{ij} = \exp -\frac{1}{2} \|c_{ij}\|_{(\mu, \Sigma)}^2$$

$$\|c\|_{(\mu, \Sigma)}^2 = (c - \mu)' \Sigma^{-1} (c - \mu)$$

μ, Σ : learnable parameters

- GRU and LSTM cells (considering that the aggregated current value of the neighbours is the previous state).

Flat graph pooling contracts the graph to a single node, by aggregating the node features with mean or max, or with a gating mechanism.



Hierarchical pooling progressively shrinks the graph,

- by selecting the most “important” nodes, where the importance is computed from the node features, directly (gPool) or with a GCN (SAGPool),
- or by collapsing nodes into “supernodes”, computed with spectral clustering (EigenPool) or with a GCN to get cluster membership probabilities (diffpool)

$$A \leftarrow S'AS$$

$$F \leftarrow S'F$$

6. GNNs can be the target of attacks (credit scoring, spam, etc.). *Evasion attacks* target a trained network; *poisoning attacks* corrupt the training data. The goal can be targeted (change the classification of a few nodes) or not (reduce the model performance). The attacker may alter the node features, add or delete nodes or edges; the attacker may have access to the model weights, or to the training data (greybox attack: they train a surrogate model and attack it), or just a black box.

White box attacks often use min-max (or bilevel) optimization, or meta-learning; black-box attacks can use reinforcement learning.

To defend against those attacks:

- Use adversarial samples during training;
- Clean the graph (“purification”), removing edges whose ends have low feature similarity, or reducing the rank of the adjacency matrix;
- Downweight suspicious nodes (replace the hidden representations with Gaussian distributions, and the the variance as a suspiciousness gauge) or penalize suspicious edges (with attention weights, trained on a clean graph to which you apply an adversarial attack, so that you know which edges are adversarial);
- Jointly learn the purification (a new adjacency matrix, S , close to the original one A , low rank, with smooth features), and train a GNN.

7. To train GNNs on large networks, GraphSAGE samples the neighbours. Instead of sampling the neighbours for each node separately, sample a set of nodes N (you need to define the probability distribution) and use $\mathcal{N}_i \cap N$ instead of \mathcal{N}_i , for a whole layer (or for all layers).

Edge-based node samplers sample pairs of nodes with high influence on one another.

Random-walk-based samplers improve the connectivity.

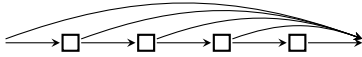
9. LSTMs can be generalized to trees or arbitrary graphs (Tree-LSTM, Graph-LSTM): just aggregate the inputs before feeding them to the LSTM cell.

To generate graphs (AE, VAE, GAN), simply generate the adjacency matrix.

10. For many NLP tasks, first process the sentence with a BiLSTM; then use a GNN on the parse tree.

Node embeddings of knowledge graphs help various downstream tasks, and knowledge graph completion.

14. To limit oversmoothing, add jump-layer connections,



randomly drop edges, or add regularization terms to force the representations of distant nodes to be distinct.

Self-supervised learning on graphs can predict:

- Node properties (degree, centrality, clustering coefficient, features);
- Pairwise similarity or ranking of node properties;
- Edge presence (if some were removed before training);
- Pairwise distances;
- Context (word2vec).

A fast and high quality multilevel scheme for partitioning irregular graphs
G. Karypis and V. Kumar (1998)

Multilevel partitioning (Metis):

- First reduces the size of the graph (coarsening) by collapsing some edges;
- Then partitions (clusters) it, using an algorithm of your choice;
- Finally refines the solution back to the original graph (Kernigham-Lin).

A fast kernel-based multilevel algorithm for graph clustering
I. Dhillon et al. (2005)

To cluster the nodes of a graph, use (weighted) k -means with kernel matrix $\sigma I + A$, $\sigma I - D + A$ or $\sigma D^{-1} + D^{-1}AD^{-1}$ (and $w = 1$ or $w = \text{degree}$).

To coarsen a graph (prior to clustering), visit the vertices in a random order:

- If x is marked, continue;
- If x is unmarked and all its neighbours are marked, mark it and continue;
- Otherwise, merge x with the vertex y maximizing

$$\frac{e_{xy}}{w_x} + \frac{e_{xy}}{w_y}$$

and mark both x and y .

This can also be used to order the nodes, when plotting the adjacency matrix.

Weighted graph cuts without eigenvectors: a multilevel approach
I.S. Shillon et al. (2007)

More details on the `gracclus` algorithm.

Simplifying graph convolutional networks
F. Wu et al. (2019)

Remove the nonlinearities and collapse the weight matrices between consecutive layers:

$$Y = \text{softmax}(S^k X W)$$

$$S = D^{-1/2}(A + I)D^{-1/2}$$

Knowledge graphs: a guided tour
A. Hogan

General $E(2)$ -equivariant steerable CNNs
M. Weiler and G. Cesa (2019)

Consider a Euclidean graph $G \leq E(2) = (\mathbf{R}, +) \rtimes O(2)$. A *steerable feature field* is defined by a feature field and its type

$$f : \mathbf{R}^2 \rightarrow \mathbf{R}^c \quad \text{feature field}$$

$$\rho : G \rightarrow \text{GL}(\mathbf{R}^c) \quad \text{representation.}$$

For instance, for scalar fields, or RGB fields, $\rho = 1$ (trivial representation), but for vector fields, $\rho(t, g) = g$ is another meaningful choice.

Equivariant maps are convolutions with a *steerable* kernel $k : \mathbf{R}^2 \rightarrow \mathbf{R}^{c_{\text{out}} \times c_{\text{in}}}$

$$\forall g \in G \forall x \in \mathbf{R}^2 \quad k(gx) = \rho_{\text{out}}(g)k(x)\rho_{\text{in}}(g^{-1}).$$

(To solve those constraints, first decompose ρ_{out} and ρ_{in} into irreducible components.)

You cannot use arbitrary nonlinearities (they have to be equivariant as well):

- For unitary representations, nonlinearities acting only on the norm $x \mapsto \sigma(x) \cdot x/\|x\|$ are equivariant:
 - NormReLU, $\sigma(u) = \text{ReLU}(u - b)$
 - Squashing, $\sigma(u) = u^2/(u^2 + 1)$
 - Gating $\sigma(u) = u/(1 + e^{-s(x)})$, where s is a scalar field;

- For regular representations (representations on $\mathbf{R}^{|G|}$, permuting the axes), or quotient representations (regular representations of G/H), coordinatewise nonlinearities are equivariant.

Implementation in `e2cnn`.

**Overcoming oversmoothness
in graph convolutional networks
via hybrid scattering networks**
F. Wenkel et al.

The *lazy random walk matrix* (the random walk matrix is WD^{-1}) $P = \frac{1}{2}(I + WD^{-1})$

$$(PX)_i = \frac{1}{2}X_i + \sum_{j \sim i} \frac{X_j}{d_j}$$

defines wavelets

$$\begin{aligned}\Psi_0 &= I = P \\ \Psi_k &= P^{2^{k-1}} - P^{2^k}\end{aligned}$$

and the *geometric scattering transform*

$$U : x \mapsto (\Psi_{k_m} \circ \sigma \circ \dots \circ \Psi_{k_2} \circ \sigma \circ \Psi_{k_1})(x).$$

Combine low-pass layers $X \mapsto \sigma(AX\Theta + B)$, band-pass layers $X \mapsto \sigma(U(X\Theta) + B)$, aggregation (or concatenation, or attention) and *graph residual convolution*

$$X \mapsto [(1 - \lambda)I + \lambda WD^{-1}]X\Theta + B$$

to limit oversmoothing and underreaching in GNNs.

**Scattering GCN: overcoming oversmoothness
in graph convolution networks**
Y. Min et al. (2020)

**Can hybrid geometric scattering networks help
solve the maximum clique problem?**
Y. Min et al. (2022)

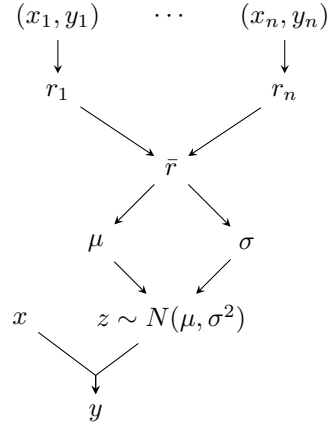
Use a scattering GCN to predict which nodes are in the maximum clique, using eccentricity, clustering coefficient, and logarithm of degree as node features, and

$$L(p) = -p'Wp + \beta p'\bar{W}p$$

as loss, where \bar{W} is the adjacency matrix of the complement of G (same node, non-edges as edges).

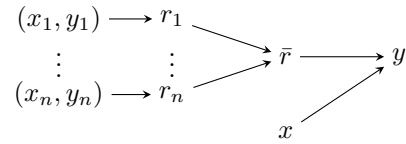
Neural processes
M. Garnelo et al. (2018)

Neural networks can replace Gaussian processes.



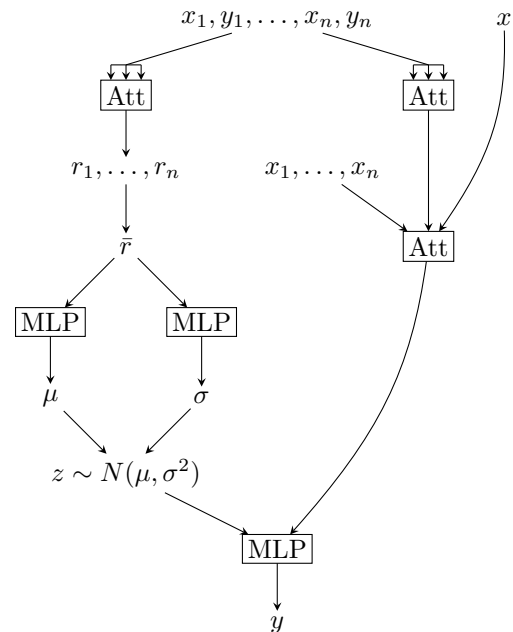
Conditional neural processes
M. Garnelo et al.

Neural process without sampling (a form of meta learning), with applications to image completion.



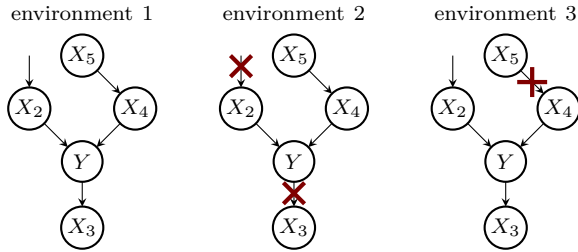
Attention neural processes
H. Kim et al.

Neural processes underfit the training data, because the average of the latent, representations gives the same weight to all context points. Try with multihead attention blocks.



**Causal inference using invariant prediction:
identification and confidence intervals**
J. Peters et al.

To distinguish between causal and non-causal models (to explain a variable Y), remember that causal models are robust to interventions: they should remain the same when estimated in different settings (interventions, regimes); we do not need to know which variables are intervened on.

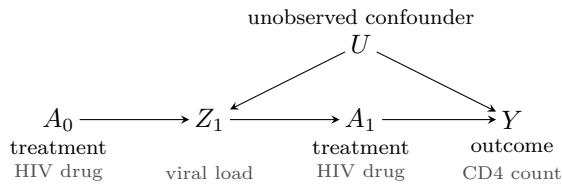


An introduction to g methods
A.I. Naimi et al. (2016)

G methods (G-formula, marginal structural models, and structural nested models) compute

$$E[Y | \text{do}(A_0 = A_1 = 1)] - E[Y | \text{do}(A_0 = A_1 = 0)]$$

in the following situation



If we only wanted $\text{do}(A_1)$, we would condition on the confounder Z_1 , but if we also want $\text{do}(A_0)$, we cannot condition on Z_1 because it is a collider. The *G-formula* is obtained by writing down the joint probability.

$$P(y, a_1, z_1, a_0) = P(y | a_1, z_1, a_0) P(a_1 | z_1, a_0) P(z_1 | a_0) P(a_0)$$

$$E[Y] = \sum_{a_1, z_1, a_0} E[Y | a_1, z_1, a_0] P(a_1 | z_1, a_0) P(z_1 | a_0) P(a_0)$$

$$E[Y | \text{do}(a_0, a_1)] = \sum_{z_1} E[Y | a_1, z_1, a_0] P(z_1 | a_0)$$

Marginal structural models use inverse probability weighting to fit the regression $y \sim a_1 + a_2$.

$$P[a_0, a_1 | z_1] = P[a_0 | a_1, z_1] P[a_1 | z_1]$$

$$= P[a_0] P[a_1 | z_1]$$

$$= \frac{1}{2} P[a_1 | z_1]$$

$$P[a_1, a_2] = \frac{1}{2} P[a_1]$$

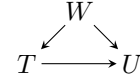
$$w_{a_0, a_1, z_1} = \frac{P[a_0, a_1]}{P[a_1, a_1 | z_1]}$$

**Estimation of the causal effects
of time-varying exposures**
K.M. Robins and M.A. Hernán

More details on *G*-methods.

**A causal analysis of market contagion:
a double machine learning approach**
J. Simonian

Double machine learning



$$\text{lm}(\text{res}(Y \sim W) \sim \text{res}(T \sim W))$$

(do the inner and outer regressions on different datasets) for

- Treatment: S&P 500 weekly returns
- Outcome: some non-US market weekly returns (for the same or next week)
- Confounders: weekly returns of other non-US markets.

**Unsuitability of NOTEARS
for causal graph discovery**
M. Kaiser and M. Sipos (2021)

NOTEARS $h(w) = \text{trace}(e^{W \odot W}) - d$ is not scale-invariant (for weighted adjacency matrices). GOLEM has the same problem.

**DAGMA: learning DAGs via M-matrices
and a log-determinant acyclic characterization**
K. Bello et al. (2022)

The NOTEARS condition, $\text{Tr}(e^{W \odot W}) = d$, for an adjacency matrix W to be that of a DAG, or

$$\text{Tr} \left(I + \frac{1}{d} W \odot W \right)^d = d,$$

can be replaced with $h(W) = 0$ or, equivalently, $\nabla h(W) = 0$, where

$$h(W) = -\log \det(sI - W \odot W) + d \log s,$$

for $s > 0$. In addition, $h(W) \geq 0$,

$$\nabla h(W) = 2(sI - W \odot W)^{-\top} \odot W.$$

Contrary to NOTEARS (where the contribution of large cycles vanishes quickly), it can detect large cycles.

**On the role of sparsity and DAG constraints
for learning linear DAGs**
I. Ng et al. (2020)

The NOTEARS hard constraint $h(B) = \text{tr}(e^{B \odot B}) - d = 0$ can be replaced with a soft constraint (GOLEM)

$$\text{Minimize}_B \mathcal{L}(B, x) + \lambda_1 \|B\|_1 + \lambda_2 h(B)$$

where the log-likelihood is

$$\mathcal{L}(B, x) = \frac{1}{2} \sum_i \log \sum_k (x_{ki} - B'_i x_k)^2 - \log |\det(I - B)|$$

(but it is not an exact characterization).

**DAG-GNN: DAG structure learning
with graph neural networks**
Y. Yu et al. (2019)

Given the adjacency matrix of a DAG, a SEM (structural equation model) is of the form $X = A'X + Z$, $Z \sim \text{noise}$, i.e., $X = (I - A')^{-1}Z$. It can be made non-linear, and the model can be extended to a VAE.

$$Z \xrightleftharpoons[f_4]{f_1} \cdot \xrightleftharpoons[(I - A')^{-1}]{(I - A')^{-1}} \cdot \xrightleftharpoons[f_3]{f_2} X$$

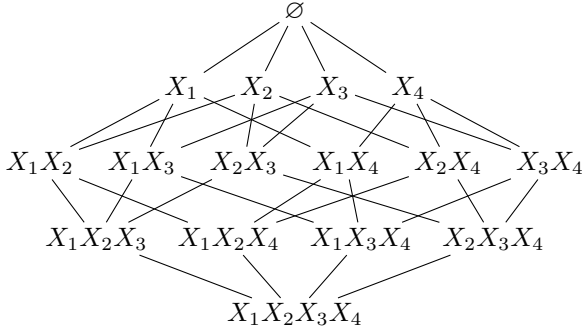
$$X = f_2[(I - A')^{-1}f_1(Z)]$$

$$Z = f_4[(I - A')^{-1}f_3(X)]$$

To ensure acyclicity: $\text{tr}(I + \alpha A \odot A)^d = d$.

**Learning optimal Bayesian networks:
a shortest path perspective**
C. Yuan and B. Malone (2013)

To find the best Bayesian network for a given dataset, it suffices to find the best leaf (childless variable) and use dynamic programming. The resulting order is a path in the *Hasse diagram* of subsets of variables. Use A^* to restrict the search to promising paths



**Improving causal discovery
by optimal Bayesian network learning**
N.Y. Lu et al. (2021)

Triplet A^* speeds up A^* -based causal discovery by restricting the search for $\text{Pa}(x)$ to subsets of neighbours of $y-x-z$ (given a PC-based initial guess of parents and children of each node).

**Domain-knowledge
in A^* -based causal discovery**
S. Kleinegesse et al.

Add domain knowledge ($x \in \text{Pa}(y)$ or $x \notin \text{Pa}(y)$) to A^* score-based causal discovery.

**Reliable causal discovery
with improved exact search
and weaker assumptions**
I. Ng et al. (2021)

Speed up A^* -based causal discovery by limiting the size of the parent graphs (A^* superstructure) and/or limiting the search to the 2-hop neighbourhood of each variable in its graphical lasso graph (local A^*).

**A neural network based model
for multi-dimensional nonlinear
Hawkes processes**
S. Joseph and S. Jain

Neural network to estimate (MLE) the kernel of a Hawkes process.

**Feature selection with annealing
for computer vision and big data learning**
A. Barbu et al.

Recursive feature elimination (RFE) alternates between

- Training an SVM model;
- Discarding the features with the smallest coefficients.

Feature selection with annealing (FSA) alternates between

- Gradient descent step;
- Discarding the features with the smallest coefficients.

Also check Boruta.

Quantile-parametrized distributions
T.W. Keelin and B.W. Powley (2011)

A quantile-parametrized distribution (QPD) is defined by

$$F^{-1}(y) = \sum_{i=1}^n a_i g_i(y), \quad y \in (0, 1).$$

For instance, with $n = 4$ and (Q -normal)

$$g_1(y) = 1$$

$$g_2(y) = \Phi^{-1}(y)$$

$$g_3(y) = y\Phi^{-1}(y)$$

$$g_4(y) = y.$$

Then, $f(x) = (\sum a_i g'_i(y))^{-1}$ where $x = F^{-1}(y)$.

Find a from n probability-value pairs $(x_i, y_i)_{1 \leq i \leq n}$ as $a = Y^{-1}x$, $Y_{ij} = g_j(y_i)$, provided Y is invertible and $\forall y \in (0, 1) \sum a_i g'_i(y) > 0$.

***Ontology development 101:
a guide to creating your first ontology***
N.F. Noy and D.L. McGuinness

An *ontology* is a formal specification of the terms used in a domain, their relations, and properties. It consists of:

- Classes (*concepts*),
- Hierarchical relations between those classes,
- *Instances* of those classes,
- Properties (*slots*) of those classes, or of the instances, each with a set of allowable values (*facets*: class, instance, symbol, free text, list, etc.).

***Axiomatic characterization
of pointwise Shapley decompositions***
M.C. Christansen (2023)

The Shapley values decompose a function $F : \{0, 1\}^d \rightarrow \mathbf{R}$ into a sum of contributions of each input variable. It can be generalized to functions with real arguments (*average sequential decomposition*, aka *pointwise Shapley values*) by writing (assuming $F(0) = 0$)

$$\begin{aligned} F(x) &= F(x) - F(0) \\ &= F(x_1, \dots, x_d) - F(x_1, \dots, x_{d-1}, 0) + \\ &\quad F(x_1, \dots, x_{d-1}, 0) - F(x_1, \dots, x_{d-2}, 0, 0) + \\ &\quad \dots \end{aligned}$$

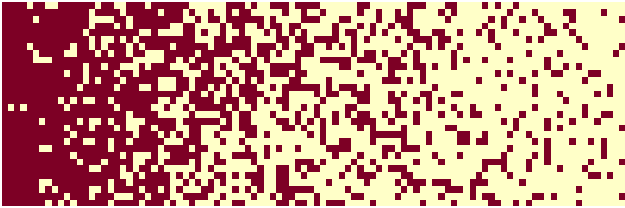
and averaging over all permutations of (x_1, \dots, x_d) .

***Perceptually uniform color space
for image signals including
high dynamic range and wide gamut***
M. Safdar et al. (2017)

$J_z A_z B_z$ is more perceptually uniform than CIELAB (for reflective colours), CIELUV (self-illuminant colours), IPT, CAM16-UCS, $IC_T C_P$; all have 3 coordinates: lightness, red-green, yellow-blue.

***Nonparametric Bayesian deep networks
with local competition***
K.P. Panousis et al. (2019)

To automatically prune a neural network, use a Bayesian neural net, with latent Bernoulli variables indicating which weights should be kept, following an Indian buffet process (IBP)



$$\begin{aligned} u_k &\sim \text{Beta}(\alpha, 1) \\ \pi_k &\sim \prod_{1 \leq i \leq k} u_i \\ z_{ik} &\sim \text{Bernoulli}(\pi_k) \end{aligned}$$

Efficient few-shot learning without prompts
L. Tunstall et al.

Before training a classifier head, fine-tune the model (in a siamese manner, with a triplet loss). Use with MPNet (a sentence transformer, trained with a paraphrase dataset, to recognize semantic similarity).

***Wasserstein-Kelly portfolios:
a robust data-driven solution
to optimize portfolio growth***
J.Y.M. Li (2023)

The Kelly portfolio

$$\begin{aligned} \text{Find} & \quad w \\ \text{To maximize} & \quad \mathbb{E}_R \log(1 + w'R) \\ \text{Such that} & \quad w \geq 0, w'1 = 1 \end{aligned}$$

where R are the asset simple returns, can be made *distributionally robust* by considering the Wasserstein ball of log-returns centered on the empirical log-return distribution (with the *simple* return ball, the optimization problem is unbounded).

$$\begin{aligned} \text{Find} & \quad w \\ \text{To maximize} & \quad \inf_{F \in \mathcal{B}} \mathbb{E}_{r \sim R} \log w'e^r \\ \text{Such that} & \quad w \geq 0, w'1 = 1 \\ \text{Where} & \quad \mathcal{B} = \{F : W_p(F, F_r) \leq \varepsilon\} \end{aligned}$$

This can be solved with `cvxpy`, after reformulation.

***Probabilistic forecasting with factor quantile
regression: application to electricity trading***
L. Maciejowska et al. (2023)

Given several forecasts $\hat{y}_1, \dots, \hat{y}_d$ (different models, or the same model estimated on different windows), forecast quantiles of y with a linear regression $\hat{y}_\alpha \sim \hat{y}_1 + \dots + \hat{y}_d$ or a principal component regression.

***An adaptive volatility method for probability
forecasting and its application to the M6
financial forecasting competition***
J. de Vilmares and N. Werge (2023)

The adaptive volatility model (AdaVol) is a non-stationary GARCH-like model.

$$\begin{aligned} \varepsilon_t &: \text{log-returns} \\ \gamma_t &: \text{sample volatility of } \varepsilon \\ \sigma_t &: \text{target volatility of } \varepsilon \text{ (unobserved)} \\ \eta_t &\sim N(0, 1) \\ \varepsilon_t &= \sigma_t \eta_t \\ \sigma_t^2 - \gamma_{t-1}^2 &= \sum_{i=1}^p \alpha_i (\varepsilon_{t-i}^2 - \gamma_{t-1}^2) + \sum_{j=1}^q \beta_j (\sigma_{t-j}^2 - \gamma_{t-1}^2) \end{aligned}$$

Performance attribution of machine learning methods for stock return prediction
S. Daul et al. (2021)

Decompose the forecasts (and the performance) of a forward-return predicting model (and the corresponding long-short decile portfolio) into:

- Linear marginal,
- Nonlinear marginal,
- Interactions.

The nonlinear marginal part has little predictive power.

Addendum on how many times cointelated pairs cross paths

B. Mahdavi-Damghani and S. Roberts (2020)

The *cointelation* model allows for arbitrary short-term correlation (even $\rho = -1$) and mean reversion ($\rho = +1$ in the long term).

$$\begin{aligned}\frac{dX}{X} &= \mu dt + \sigma dW_1 && \text{leading} \\ dY &= \theta(X - Y)dt + \sigma Y dW_2 && \text{lagging} \\ d\langle W_i, W_2 \rangle &= \rho dt\end{aligned}$$

Exploring the advantages of transformers for high-frequency trading
F. Barez et al. (2023)

Forecast the future (100ms) log-returns with a transformer encoder, linear decoder, spiking activation

$$S(x) = \begin{cases} x & \text{if } x \geq \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

with `pytorch-spiking` (and PReLU) and quantile loss.

Reinforcement learning for trading systems and portfolios
J. Moody and M. Saffell (1998)

Train your trading/investment strategy end-to-end, maximizing the Sharpe ratio, instead of minimizing the sum of squared residuals.

Macroeconomic forecasting and sovereign risk assessment using deep learning techniques
A. Petropoulos et al.

Predict 9 monthly macroeconomic variables from their 1-year lags and 3 contemporaneous variables, with

- Bayesian model averaging (BMA);
- MLP (5 layers, ReLU, dropout);
- Bayesian neural net;
- Bayesian LWTA (local winner takes all) neural net.

Deep adaptive input normalization for time series forecasting
N. Passalis et al.

Before processing a time series with a neural net, DAIN normalizes it, adaptively.

$$\begin{aligned}a &\leftarrow \text{LSTM}(x) \\ x &\leftarrow x - Wa \\ b &\leftarrow \text{LSTM}(x) \\ x &\leftarrow x \oslash Wb \\ c, d &\leftarrow \text{LSTM}(x) \\ x &\leftarrow x \odot \text{sigm}(Wc + d)\end{aligned}$$

Expectile hidden Markov regression models for analyzing cryptocurrency returns
B. Foroni et al. (2023)

Expectile regression replaces the squares in the loss function with

$$\omega_\tau(u) = \begin{cases} \tau u^2 & \text{if } u \geq 0 \\ (1 - \tau)u^2 & \text{if } u < 0, \end{cases}$$

where $\tau \in (0, 1)$.

$$\underset{\beta}{\text{Minimize}} \mathbb{E}[\omega_\tau(y - X'\beta)]$$

It is often estimated with IRLS (iteratively reweighted least squares).

The *asymmetric normal* distribution has density

$$f(y) \propto \exp -\omega_\tau \left(\frac{y - \mu}{\sigma} \right).$$

The asymmetric normal hidden Markov model (HMM) can be estimated with the expectation-maximization (EM) algorithm.

Frog in the pan: continuous information and momentum
Z. Da et al. (2014)

Investors are inattentive to information arriving continuously in small amounts: combine/replace momentum with/by the “information discreteness”,

$$\text{ID} = \text{sign}(\text{momentum}) \times (\% \text{neg} - \% \text{pos}).$$

Quantum computing: lecture notes
R. de Wolf (2022)

Quantum computing is linear algebra on the Hilbert space $H = (\mathbf{C}^2)^{\otimes n}$. The standard basis of \mathbf{C}^2 is denoted

$$\begin{aligned}|0\rangle &= e_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ |1\rangle &= e_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.\end{aligned}$$

The coordinates in that basis are called *amplitudes* (but these are complex numbers). The corresponding row

vectors (conjugate transpose) are denoted $\langle 0|$, $\langle 1|$. Another basis is

$$\begin{aligned}|+\rangle &= (|0\rangle + |1\rangle)/\sqrt{2} \\ |-\rangle &= (|0\rangle - |1\rangle)/\sqrt{2}.\end{aligned}$$

A *classical state* is an element of the standard basis. A pure quantum *state* is a vector of norm 1, up to a scalar multiple; alternatively, it is a 1-dimensional subspace, or a projection onto a 1-dimensional subspace. A quantum state is a *superposition* of classical states, i.e., a linear combination of basis vectors. A *mixed quantum state* is a probability distribution over pure quantum states, $\rho = \sum p_i |\phi_i\rangle \langle \phi_i|$. An *entangled state* in $\mathbf{C}^2 \otimes \mathbf{C}^2$ is a state which is not of the form $u \otimes v$ (elements of $\mathbf{C}^2 \otimes \mathbf{C}^2$ are linear combinations of such *product states*), e.g., the *EPR pair*

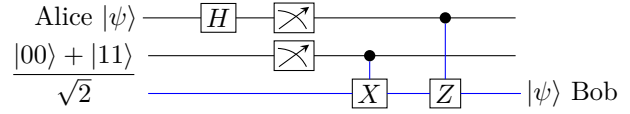
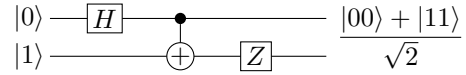
$$\frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

A *measurement* of a quantum state $|\phi\rangle = \sum \alpha_i |i\rangle$ in the standard (or “computational”) basis ($|0\rangle, |1\rangle, \dots, |N-1\rangle$) replaces $|\phi\rangle$ with the classical state $|i\rangle$ with probability $\|\alpha_i\|^2$. A *projective measurement* of $|\phi\rangle$ wrt projections P_1, \dots, P_m such that $\sum P_i = I$ replaces $|\phi\rangle$ with (a state proportional to) $P_i |\phi\rangle$ with probability $\langle \phi | P_i | \phi \rangle = \text{tr } P_i |\phi\rangle \langle \phi|$. A *POVM* (positive operator-valued measurement) wrt positive semi-definite matrices E_1, \dots, E_m such that $\sum E_i = I$ replaces $|\phi\rangle$ with a post-measurement state associated with E_i (not defined) with probability $\text{tr } E_i |\phi\rangle \langle \phi|$.

States can be transformed with unitary matrices U , or *gates*, $|\phi\rangle \mapsto U |\phi\rangle$ (or $\rho \mapsto U \rho U^\dagger$), for instance

$$\begin{aligned}I &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ X &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} && \text{bitflip (NOT)} \\ Y &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ Z &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} && \text{phase flip} \\ H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 1 & -1 \end{pmatrix} && \text{Hadamard} \\ R_\phi &= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} && \text{phase gate} \\ T &= R_{\pi/4} \\ \text{CNOT} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}\end{aligned}$$

The CNOT (controlled NOT) gate applies a bitflip X (NOT) to the second qubit if the first is $|1\rangle$; the Toffoli gate (CCNOT) is similar, but has two control qubits: if they are both 1, the third qubit is flipped. The composition of those gates is often depicted as a quantum circuit.

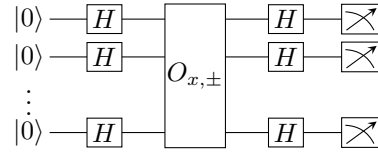


The *query* for $x \in \{0, 1\}^N$ is the unitary map (a permutation matrix)

$$O_x : \begin{cases} (\mathbf{C}^2)^{\otimes n} \otimes \mathbf{C}^2 & \longrightarrow (\mathbf{C}^2)^{\otimes n} \otimes \mathbf{C}^2 \\ |i, b\rangle & \mapsto |i, b \oplus x_i\rangle \end{cases}$$

where $n = \lceil \log_2 N \rceil$. The *phase query* $O_{x,\pm}$ uses $|i, -\rangle \mapsto (-1)^{x_i} |i, -\rangle$ instead.

The *Deutsch-Jozsa algorithm* tells if $x \in \{0, 1\}^N$ is constant or half 1's and half 0's.



Given $x \in \{0, 1\}^N$ such that $x_i \equiv ia \pmod{2}$, the *Bernstein-Vazirani algorithm* finds a .

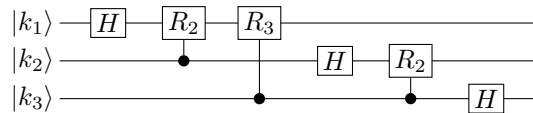
Given $x \in \{0, 1\}^N$ such that $x_i = x_j$ iff $i = j$ or $i = j \oplus s$, for some $s \in \{0, 1\}^n$, the *Simon algorithm* finds s .

The *Fourier transform*

$$(F_N)_{jk} = \frac{1}{\sqrt{N}} e^{2\pi i jk/N}$$

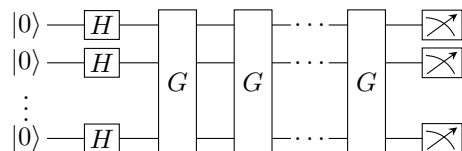
generalizes the Hadamard transform $H = F_2$. The Fast Fourier transform (FFT) computes $F_N v$ efficiently by recursively separating even and odd terms. The *Quantum Fourier transform* only requires $O((\log N)^2)$ gates, instead of $O(N \log N)$.

$$F_N |k\rangle = \bigotimes_{\ell=1}^n \frac{1}{\sqrt{2}} \left(|0\rangle + e^{2\pi i k/2^\ell} \right)$$



Shor's *period finding algorithm* is $O((\log N)^2 (\log \log N)^2 \log \log \log N)$.

Grover's search algorithm provides a quadratic speed-up.



$$G = H^{\otimes n} R_0 H^{\otimes n} O_{x,\pm}$$

Unitary operators appearing in quantum mechanics are often of the form $U = e^{-iHt}$, form some Hermitian operator H (the Hamiltonian). If $H = \sum H_j$ (where, for instance, the H_j 's only act on a small number of qubits), we would like to decompose U as a product of simpler (smaller) unitary operators, but $e^{-iHt} \neq \prod e^{-iH_j t}$ in general (if the H_j 's do not commute). The *Lie-Trotter* formula approximates U as

$$e^{-i \sum H_j t} \approx \left(\prod_j e^{i H_j t/r} \right)^r.$$

There are other Hamiltonian approximations (linear combination of unitaries (LCU), block-encoded matrices).

The *HHL algorithm* solves a *quantum* linear system $Ax = b$, *i.e.*, finds a n -qubit state close to

$$|x\rangle = \frac{1}{\|x\|} \sum_{i=0}^{N-1} x_i |i\rangle,$$

Similarly, *quantum machine learning* learns from data encoded as quantum states (the model is a quantum circuit, and the parameters are the gate parameters, *e.g.*, θ for R_θ).

Error correction is more necessary than with classical computing, but possible.

Quantum mechanics for mathematicians A. Alekseev (2019)

1. A finite-dimensional **quantum system** is defined by (\mathcal{H}, H) :

- A1 A Hilbert space over \mathbf{C} , \mathcal{H} ;
- A2 The set of *observables*,

$$\mathcal{A} = \{A \in \text{End } \mathcal{H} : A^* = A\}$$

(note that \mathcal{A} is a Lie algebra);

- A3 The set of *states*

$$\mathcal{S} = \{M \in \text{End } \mathcal{H} : M^* = M, \text{tr } M = 1, M \geq 0\};$$

a *pure state* is a projection in a 1-dimensional subspace; pure spaces form the projective space $\mathbf{P}(\mathcal{H})$;

- A4 A measurement map

$$\begin{cases} \mathcal{A} \times \mathcal{S} & \longrightarrow \{\text{probability measures on } \mathbf{R}\} \\ (A, M) & \longmapsto \mu_A^M \end{cases}$$

defined by

$$\mu_A^M(E) = \text{tr}(PP_A(E)) = \sum_{i: \lambda_i \in E} \text{tr}(MP_i)$$

$$A = \sum \lambda_i P_i$$

P_i : projections on the eigenspaces

$$P_A(E) = \sum_{i: \lambda_i \in E} P_i$$

P_A : projection-valued measure

For a pure state $M = P_u$, $\|u\| = 1$, and 1-dimensional eigenspaces

$$\begin{aligned} \mu_A^M &= \sum_i \text{tr}(MP_i) \delta_{\lambda_i} \\ &= \sum_i |\langle e_i, u \rangle|^2 \delta_{\lambda_i} \\ &= \sum_i p_i \delta_{\lambda_i}; \end{aligned}$$

the $\langle e_i, u \rangle$ are called *probability amplitudes* (without the modulus). The mean and variable of an observable A are

$$\bar{A} = \int_{\mathbf{R}} \lambda d\mu_A^M = \sum \lambda_i p_i$$

$$\text{Var}[A] = \int_{\mathbf{R}} (\lambda - \bar{A})^2 d\mu_A^M = \sum (\lambda_i - \bar{A})^2 p_i.$$

More generally (“functional calculus”), for a function $f : \mathbf{R} \rightarrow \mathbf{R}$, $f(A) = \sum f(\lambda_i) P_i$ is still self-adjoint and

$$\overline{f(A)} = \int_{\mathbf{R}} f(\lambda) d\mu_A^M = \sum f(\lambda_i) p_i.$$

A5 $H \in \mathcal{A}$, the *Hamiltonian* of the system; its eigenvalues are denoted E_1, \dots, E_n ; the evolution of the system is then defined by

$$\frac{dM(t)}{dt} = -\frac{i}{\hbar} [H, M].$$

It is often written for pure states P_u , *i.e.*, projections on 1-dimensional subspaces,

$$\frac{du(t)}{dt} = -\frac{i}{\hbar} H u(t).$$

It can be solved with the (unitary) *evolution operator* $U(t) = \exp(-\frac{i}{\hbar} H t)$ as $u(t) = U(t)u_0$ (for general states, $M_t = U_t M_0 U_t^{-1}$). In the Heisenberg picture, the state is constant and the observables change

$$\frac{dA(t)}{dt} = +\frac{i}{\hbar} [H, A].$$

The Heisenberg uncertainty principle states

$$\sigma_{A_1}^2 \sigma_{A_2}^2 \geq \frac{1}{2} |\text{tr } M([A_1, A_2])|^2.$$

An observable $A \in \mathcal{A}$ is a *conserved quantity*, or a *quantum integral* if $[H, A] = 0$.

The Lie algebra of symmetries is

$$\text{sym } H = \{G \in i\mathcal{A} : [B, H] = 0\} \subset i\mathcal{A}.$$

The group of symmetries is

$$\text{Sym } H = \{g \in U(\mathcal{H}) : g H g^{-1} = H\}$$

A set of observables (H, A_1, \dots, A_m) is a quantum *integrable system* if they commute and their symmetry group $\text{Sym}(H, A_1, \dots, A_m)$ is as small as possible, *i.e.*, $U(1) \times \dots \times U(1)$, *i.e.*, the joint eigenspaces are of dimension 1. Given a basis of common eigenvectors $(e_a)_{1 \leq a \leq n}$,

$$\begin{aligned} H e_a &= E_a e_a \\ A_i e_a &= \alpha_a^i e_a \end{aligned}$$

there is a bijection between the set of *quantum numbers* $\{(E_a, \alpha_a^1, \dots, \alpha_a^m) : 1 \leq a \leq m\}$ and the basis. Examples include:

- The spin $\frac{1}{2}$ particle, $\mathcal{H} = \mathbf{C}^2$, with the Pauli matrices $\sigma_1, \sigma_2, \sigma_3$ as an \mathbf{R} -basis of \mathcal{A} and, for instance,

$$H = \begin{pmatrix} \alpha & 0 \\ 0 & -\alpha \end{pmatrix};$$
- The Heisenberg magnetic chain, $\mathcal{H} = (\mathbf{C}^2)^{\otimes N}$,

$$H = J \sum_{i=1}^N \sum_{a=1}^3 \sigma_a^i \sigma_a^{i+1}$$

where $\sigma_a^i = 1 \otimes \dots \otimes 1 \otimes \sigma_a \otimes 1 \otimes \dots \otimes 1$. Note that $L_a = \sum_i \sigma_a^i$ are conserved quantities, but there are many more.

2. Given a (possibly unbounded) operator on a Hilbert space \mathcal{H}

$$A : \mathcal{H} \supset D(A) \longrightarrow \mathcal{H}$$

with dense domain $D(A)$, its *adjoint* is $A^* : D(A^*) \rightarrow \mathcal{H}$ where

$$D(A^*) = \{v \in \mathcal{H} : u \mapsto \langle u, Au \rangle \text{ is bounded on } D(A)\}$$

and A^* is given by the Riesz representation theorem $\langle v, Au \rangle = \langle A^* v, u \rangle$. The domain $D(A^*)$ need not be dense.

An operator is *closed* if its graph is closed; it is *closable* if it admits a closed extension; it is *symmetric* if $\forall u, v \in D(A), \langle Au, v \rangle = \langle u, Av \rangle$. For a symmetric closable operator, $A \subset \bar{A} \subset A^*$. An operator is *self-adjoint* if $A^* = A$ (in particular, $D(A) = D(A^*)$); it is *essentially self-adjoint* if $\bar{A} = A^*$.

If A is symmetric,

$$A^* = A \text{ iff } \text{Im}(A + i) = \text{Im}(A - i) = \mathcal{H}.$$

If A is symmetric and closed, it admits a self-adjoint extension iff $n_+(A) = n_-(A)$; its self-adjoint extensions are in bijection with unitary operators

$$U : \ker(A^* - i) \rightarrow \ker(A^* + i).$$

Given two self-adjoint operators A, B , with $D(A) \subset D(B)$, if $\forall u \in D(A) \ \|Bu\| \leq \alpha \|Au\| + \beta \|u\|$, for some $\alpha < 1$, then $A + B : D(A) \rightarrow \mathcal{H}$ is self-adjoint (Kato-Rellich).

In ℓ^2 , the operator $Ae_i = a_i e_i$, on

$$D(A) = \{\text{sequences eventually zero}\}$$

is symmetric if the a_i 's are real; its adjoint is defined on

$$D(A^*) = \{v : \sum a_i^1 \|v_i\|^2 < \infty\};$$

it is essentially self-adjoint: $\bar{A} = A^*$.

On $L^2([0, 1])$, the momentum operator

$$A = -i\hbar \frac{d}{dx},$$

defined on $\mathcal{C}_0^1([0, 1], \mathbf{C})$ (continuously differentiable functions u with $u(0) = u(1) = 0$) is symmetric but not essentially self-adjoint: $\bar{A} \subsetneq A^*$, since $\mathcal{C}^1([0, 1]) \subset D(A^*)$. Replacing the boundary conditions with $u(0) = u(1)$ (*i.e.*, replacing $[0, 1]$ with \mathbf{S}^1) fixes the problem.

In $L^2(\mathbf{R}^n)$, multiplication by a measurable function $V = V(x_1, \dots, x_n)$ is a self-adjoint operator \hat{V} on $D(\hat{V}) = \{u \in L^2(\mathbf{R}^n) : \int_{\mathbf{R}^n} V^2 |u|^2 dx < \infty\}$.

Using the Fourier transform and the Schwartz functions $\mathcal{S} = \{u \in \mathcal{C}^\infty : \forall i, j \sup_x |x^i \partial_j u| < \infty\}$ one can build self-adjoint extensions of $-i\partial_i$.

In $L^1(\mathbf{R}^3)$ (not $L^2(\mathbf{R}^n)$ for $n \geq 4$) $A = -\Delta + \hat{V}_1 + \hat{V}_2$, where $V_1 \in L^2, V_2 \in L^\infty$, is self-adjoint on $D(-\Delta)$ and bounded below ($\forall u \ \langle u, Au \rangle \geq C \|u\|^2$).

3. The **spectrum** $\sigma(A)$ of $A : D(A) \rightarrow \mathcal{H}$ is

$$\{x \in \mathbf{C} : A - x \text{ is not invertible}\};$$

it is closed. For instance,

$$\begin{aligned} A &= -i \frac{d}{dx} \\ D(A) &= \mathcal{C}^1(\mathbf{S}^1) \\ \sigma(A) &= \mathbf{Z} \end{aligned}$$

$$\begin{aligned} A &= \hat{x} \\ D(A) &= \left\{ u \in L^2(\mathbf{R}) : \int_{\mathbf{R}} x^2 |u(x)|^2 dx < \infty \right\} \\ \sigma(A) &= \mathbf{R} \end{aligned}$$

If $A : D(A) \rightarrow \mathcal{H}$ is self-adjoint, there is a unique projection-valued measure $E \mapsto P_A(E)$ on \mathbf{R} such that

$$\begin{aligned} D(A) &= \{u \in \mathcal{H} : \int_{\mathbf{R}} \lambda^2 f \langle u, P_A(\lambda) u \rangle < \infty\} \\ Au &= \int_{\mathbf{R}} \lambda d(P_A(\lambda)u) \text{ (Riemann integral)} \\ \langle u, Au \rangle &= \int_{\mathbf{R}} \lambda d \langle u, P_A(\lambda)u \rangle \end{aligned}$$

where $P_A(\lambda) = P_A((-\infty, \lambda))$.

For $f \in \mathcal{C}(\mathbf{R}, \mathbf{C})$, we can define (functional calculus) $f(A)$ as

$$\begin{aligned} D(f(A)) &= \left\{ u \in \mathcal{H} : \int_{\mathbf{R}} |f(\lambda)|^2 d \langle u, P_A(\lambda)u \rangle < \infty \right\} \\ f(A)(u) &= \int_{\mathbf{R}} f(\lambda) d(P(\lambda)u). \end{aligned}$$

In particular, for the Hamiltonian $H : D(H) \rightarrow \mathcal{H}$, we can define the 1-parameter subgroup of unitary evolution operators

$$U(t) = \exp\left(-\frac{i}{\hbar}Ht\right).$$

5. Given two self-adjoint operators A, B , with $D(A) \cap D(B)$ dense, such that $A + B$ be self-adjoint,

$$\forall \psi \quad e^{i(A+B)}\psi - (e^{iA/n}e^{iB/n})^n\psi \longrightarrow 0.$$

If, in addition, A and B are bounded below,

$$\forall \psi \quad e^{-(A+B)}\psi - (e^{-A/n}e^{-B/n})^n\psi \longrightarrow 0.$$

The *Feynman path integral* applies this to $H = A + B$, $A = -(\hbar^2/2m)\Delta$, $B = V(x)$, and conjectures (wrongly) that the resulting limit can be expressed as an integral on a space of paths.

If $A : \mathbf{R}^n \rightarrow \mathbf{R}^n$ is self-adjoint with positive eigenvalues, then the *Gaussian integral* is

$$\int_{\mathbf{R}^n} d^n x e^{-\frac{1}{2}\langle x, Ax \rangle + \langle x, y \rangle} = \frac{(2\pi)^{n/2}}{(\det A)^{1/2}} e^{\frac{1}{2}\langle y, A^{-1}y \rangle}$$

(there is a similar formula for complex exponential).

There are several generalizations of the determinant to infinite dimensions. The *Fredholm determinant* is

$$\det_F(I - \lambda A^{-1}) = \prod \left(1 - \frac{\lambda}{\lambda_k}\right) \quad \text{if } \sum \frac{1}{|\lambda_k|} < \infty$$

(it is a holomorphic function with zeroes at $\lambda = \lambda_k$. If the eigenvalues of A are positive, its zeta function is

$$\zeta_A(s) = \sum \frac{1}{\lambda_k^s}.$$

In finite dimension, $\zeta(0) = n$ and $\zeta'(0) = \log \det A$; in infinite dimension, if ζ admits an analytic continuation to a neighbourhood of 0, we can define the ζ -regularized determinant

$$\det_\zeta(A) = \exp -\zeta'_A(0)$$

and the ζ -regularized dimension $\delta_A = \zeta_A(0)$. In particular, $\det_\zeta(cA) = c^{\delta_A} \det_\zeta(A)$. Those two notions are related:

$$\det_\zeta(A - \lambda) = \det_\zeta(A) \det_F(I - \lambda A^{-1}).$$

***SDDP.jl: a Julia package
for stochastic dual dynamic programming***
O. Dowson and L. Kapelevich

A multistage stochastic optimization problem is of the form

$$\begin{array}{ll} \text{Find} & u \\ \text{To maximize} & \mathbb{E}_{\omega_1} [V_1(x_1, \omega_1)] \\ \text{Where} & x_1 = 0 \quad \text{state} \\ & x_{t+1} = f(x_t, \omega_t, u_t) \\ & \omega_t \sim p_t \quad \text{noise} \\ & V_t(x_t, \omega_t) = \max_u g(x_t, \omega_t, u) + \\ & \quad \mathbb{E}_{\omega_{t+1}} [V_{t+1}(x_{t+1}, \omega_{t+1})] \\ & h(x_{\leq t}, u_{\leq t}, \omega_{\leq t}) = 0 \quad \text{constraints} \end{array}$$

Dynamic low-rank approximation

O. Koch and C. Lubich (2007)

One can compute a low-rank approximation of a smooth path $A : [0, 1] \rightarrow \mathbf{R}^{m \times n}$ as

$$X(t) = \underset{X: \text{rank } X=k}{\text{Argmin}} \|X - A(t)\|_2^2.$$

Alternatively, try

$$\hat{Y}(t) = \underset{X: \text{rank } X=k}{\text{Argmin}} \|X - \dot{A}_t\|_2^2;$$

this gives a *smooth* low-rank approximation.

Sheaf neural networks

J. Hansen and T. Gebhart (2020)

A *cellular sheaf* \mathcal{F} on a graph $G = (V, E)$ is the datum of

- A vector space $\mathcal{F}(v)$ for each node $v \in V$;
- A vector space $\mathcal{F}(e)$ for each edge $e \in E$;
- A linear map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}v \rightarrow \mathcal{F}e$ for each incident vertex-edge pair $v \trianglelefteq e$.

A *0-cochain*, resp. *1-cochain*, is an element of

$$C^0(G, \mathcal{F}) = \prod_{v \in V} \mathcal{F}(v)$$

$$\text{resp. } C^1(G, \mathcal{F}) = \prod_{e \in E} \mathcal{F}(e).$$

The *coboundary* map is

$$\delta : \begin{cases} C^0(G, \mathcal{F}) & \longrightarrow & C^1(G, \mathcal{F}) \\ (x_v)_{v \in V} & \longmapsto & (\mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u)_{e \in E} \end{cases}$$

(for an arbitrary orientation of the edges).

The *sheaf Laplacian* is $L_{\mathcal{F}} = \delta^\top \delta$. For the *constant sheaf* $\mathcal{F}v = \mathbf{R}$, $\mathcal{F}e = \mathbf{R}$, $\mathcal{F}_{v \trianglelefteq e} = \text{Id}$, it is the standard Laplacian. With different signs for $\mathcal{F}_{v \trianglelefteq e}$, it can model *signed* graphs and heterophily. The *normalized Laplacian* is $\tilde{L}_{\mathcal{F}} = D^{-1/2} L_{\mathcal{F}} D^{-1/2}$, where $D = \text{diag } L_{\mathcal{F}}$. The corresponding *diffusion operator* is $\tilde{H}_{\mathcal{F}} = I - \tilde{L}_{\mathcal{F}}$. A *sheaf convolution layer* is

$$X \longmapsto \rho[\tilde{H}_{\mathcal{F}}((I_{|V|} \otimes B)XA)]$$

where ρ is an elementwise nonlinearity.

**Sheaf neural networks
with connection Laplacians
F. Barbero et al. (2022)**

GNNs work poorly on heterophilic data (homophily is an inductive bias) and suffer from oversmoothing. Replacing the Laplacian with a sheaf Laplacian can address those problems. Instead of choosing the sheaf manually using domain knowledge, or learning it end-to-end,

$$\mathcal{F}_{v \triangleleft (v,u)} = \Phi(x_v, x_u) \in \mathbf{R}^{d \times d},$$

compute it at preprocessing time, using local PCA:

$$\begin{aligned} i, j &: \text{ nodes} \\ \hat{X}_i &: \text{ features of the neighbours of } i \\ \hat{X}_i &= U_i \Sigma_i V_i' \text{ (SVD)} \\ O_i &= U_i[:, : d] \\ O_i' O_j &= U \Sigma V' \text{ (SVD)} \\ O_{ij} &= UV' \\ O_{ij} &= \mathcal{F}_{j \triangleleft e}^\top \mathcal{F}_{i \triangleleft e}. \end{aligned}$$

This is the parallel transport: it gives the entries of $L_{\mathcal{F}}$, up to sign for the off-diagonal blocks.

**Neural sheaf diffusion:
a topological perspective on heterophily
and oversmoothing in GNNs
C. Bodnar et al. (2022)**

A *discrete $O(d)$ -bundle* is a cellular sheaf with orthogonal maps $\mathcal{F}_{v \triangleleft e} \in O(d)$. If the *transports*

$$P_{\gamma: v_1 \rightarrow \dots \rightarrow v_n} = (\mathcal{F}_{v_{n-1} \triangleleft e}^\top \mathcal{F}_{v_n \triangleleft e}) \cdots (\mathcal{F}_{v_1 \triangleleft e}^\top \mathcal{F}_{v_2 \triangleleft e})$$

are path-independent, then there is a nontrivial global section: $H^0 \neq 0$ (in fact, $\dim H^0 = d$). Diffusion, if run long enough, yields a signal constant on each connected component. Diffusion on a well-chosen $O(d)$ -bundle can linearly separate classes, even in presence of heterophily.

Learn the sheaf to use in *neural sheaf diffusion* layers

$$X_{t+1} = X_t - \sigma[\Delta_{\mathcal{F}(t)}(I \otimes W_1^t)X_t W_2^t]$$

as

$$\mathcal{F}_{v \triangleleft (v,u)} = \Phi(x_v, x_u) = \sigma(V[x_v \| x_u]).$$

Parametrize orthogonal matrices with *Householder reflections*.

The sheaf Laplacian is a positive semi-definite block matrix; the diagonal blocks are

$$(L_{\mathcal{F}})_{vv} = \sum_{v \triangleleft e} \mathcal{F}_{v \triangleleft e}^\top \mathcal{F}_{v \triangleleft e}$$

and the off-diagonal blocks are

$$(L_{\mathcal{F}})_{uv} = -\mathcal{F}_{v \triangleleft e}^\top \mathcal{F}_{u \triangleleft e}.$$

**Efficient orthogonal parametrization
of recurrent neural networks
using Householder reflections
Z. Mhammedi et al. (2017)**

All orthogonal matrices in $O(n)$ can be written as $H_n(u_n)H_{n-1}(u_{n-1}) \cdots H_1(u_1)$ where $u_k \in \mathbf{R}^k \setminus \{0\}$, $u_1 \in \{\pm 1\}$,

$$H_k(u) = \begin{pmatrix} I_{n-1} & 0 \\ 0 & I_k - 2 \frac{uu'}{\|u\|^2} \end{pmatrix}, \quad H_1(u) = \begin{pmatrix} I & 0 \\ 0 & u \end{pmatrix}.$$

Use in RNNs to prevent vanishing or exploding gradients.

**Knowledge sheaves: a sheaf-theoretic
framework for knowledge graph embedding
T. Gebhart and J. Hansen**

A *knowledge graph schema* is a directed multigraph with entity types as nodes and relation types as edges. A *knowledge graph* (KG) is a graph morphism to a KG schema $k : G \rightarrow S$. A *knowledge sheaf* is a cellular sheaf \mathcal{F} on a KG schema; it induces a sheaf $k^* \mathcal{F}$ on the KG. A *sheaf embedding* of a KG G is a global section $x \in H^0(G, k^* \mathcal{F}) \subset C^0(G, k^* \mathcal{F})$. It can be learned, together with the sheaf \mathcal{F} , as

$$\underset{x \in C^0(G, k^* \mathcal{F})}{\text{Minimize}} \quad x' L_{k^* \mathcal{F}} x$$

with a penalty (negative examples) to avoid the zero solution (in practice, a margin ranking loss works better). Translational embeddings (TransE) are a special case.

**A brief note for sheaf structures on posets
C.S. Hu (2020)**

A *cellular sheaf* on a poset (P, \leq) is a functor $(P, \leq) \rightarrow \mathbf{Set}$. In the *Alexandrov topology* on a poset (P, \leq) , the open sets are the $U \subset P$ such that

$$\forall x \in U \quad \forall y \in P \quad x \leq y \implies y \in U.$$

The *open stars* $U_x = \{y \in P : x \leq y\}$, $x \in P$, form an open basis. The closure of $x \in P$ is $\bar{x} = \{y \in P : y \leq x\}$. A cellular sheaf \mathcal{F} on (P, \leq) induces a sheaf on P with the Alexandrov topology with $\mathcal{F}(U_x) = \mathcal{F}(x)$.

**Towards a spectral theory of cellular sheaves
J. Hansen and R. Ghrist**

A *cellular sheaf* on a *regular cell complex* is the datum of a vector space $\mathcal{F}(\sigma)$ for each cell σ of X , together with a linear transformation (restriction map) $\mathcal{F}_{\sigma \triangleleft \tau} : \mathcal{F}(\sigma) \rightarrow \mathcal{F}(\tau)$ for each incident cell pair, such that $\mathcal{F}_{\sigma \triangleleft \sigma} = \text{Id}$ and $\mathcal{F}_{\sigma \triangleleft \tau} \circ \mathcal{F}_{\tau \triangleleft \rho} = \mathcal{F}_{\sigma \triangleleft \rho}$. The space of *cochains* is

$$C^k(X, \mathcal{F}) = \bigoplus_{\dim \sigma = k} \mathcal{F}(\sigma)$$

and the coboundary $\delta^k : C^k \rightarrow C^{k+1}$ is

$$\delta^{k+1}|_{\mathcal{F}(\sigma)} = \sum_{\dim \tau = k+1} [\sigma : \tau] \mathcal{F}_{\sigma \triangleleft \tau}$$

where the *signed incidence relation* satisfies

- $\sigma \triangleleft \tau \implies \sum_{\gamma} [\sigma : \gamma][\gamma : \tau] = 0$;
- $[\sigma : \tau] \neq 0$ only if $\sigma \triangleleft \tau$ and there are no cells between σ and τ .

The *Hodge Laplacian* is $\Delta = (\delta + \delta^*) = \delta^* \delta + \delta \delta^*$. Δ^k can be decomposed as $\Delta^k = \Delta_+^k + \Delta_-^k$ with $\Delta_+^k = (\delta^k)^*(\delta^k)$ and $\Delta_-^k = (\delta^{k-1})(\delta^{k-1})^*$. The cohomology is then

$$H^k := \frac{\ker \delta^k}{\text{Im } \delta^{k-1}} = \ker \Delta^k.$$

The degree-zero Laplacian generalizes the graph Laplacian; its diagonal and off-diagonal blocks are

$$\begin{aligned} \Delta_{vv}^0 &= \sum_{v \triangleleft e} \mathcal{F}_{v \triangleleft e}^* \mathcal{F}_{v \triangleleft e} \\ \Delta_{uv}^0 &= -\mathcal{F}_{u \triangleleft e}^* \mathcal{F}_{v \triangleleft e}. \end{aligned}$$

The Laplacian of a *normalized sheaf*

$$\forall x, y \in \mathcal{F}(\sigma) \cap (\ker \delta)^\perp \quad \langle \delta x, \delta y \rangle = \langle x, y \rangle$$

generalizes the normalized graph Laplacian.

Path homologies of deep forward networks **S. Chowdhury et al.**

An *elementary p-path* in a graph (V, E) is a sequence of $p + 1$ nodes (x_0, \dots, x_p) ; it is *allowed* if $\forall i (x_i, x_{i+1}) \in E$. Let $\Lambda_p(V)$ and $A_p(V)$ be the free vector spaces generated by elementary and allowed paths. The boundary of a path is $\partial_p[x_0, \dots, x_p] = \sum (-1)^i [x_0, \dots, \hat{x}_i, \dots, x_p] \in \Lambda_p(V)$. The ∂ -invariant paths $\Omega_p = \{c \in A_p : \partial_p c \in A_{p-1}\}$ form a chain complex

$$\xrightarrow{\partial_3} \Omega_2 \xrightarrow{\partial_2} \Omega_1 \xrightarrow{\partial_1} \Omega_0 \xrightarrow{\partial_0} k \longrightarrow 0$$

defining the *path homology* $H_p^\pm(G) = \ker \partial_p / \text{Im } \partial_p$. The *directed flag complex* (DFC) is defined from sequences (x_0, \dots, x_p) satisfying $\forall i < j (x_i, x_j) \in E$.

Discovering governing equations from data by sparse identification of nonlinear dynamical systems **S.L. Brunton et al.**

To learn a differential equation, *i.e.*, find f in $\dot{x}(t) = f(x(t))$ from trajectories x , SINDy:

- Estimates \dot{x} , with total variation regularization to denoise it;
- Builds a library of nonlinear transformations of the columns of x ;
- Uses a sparse regression.

Learning ODE models with qualitative structure using Gaussian processes **S. Ridderbusch et al.**

To learn a differential equation $\dot{x}(t) = f(x(t))$, model (x, \dot{x}) with a Gaussian process (this gives a less noisy estimate of \dot{x} , with a matrix kernel (uncoupled separable matrix (USM) kernel); to account for symmetries, use group integration matrix (GIM) kernels.

Statistical inference on random dot product graphs: a survey **A. Athreya et al. (2017)**

A *random dot product graph* (RDPG) is an Erdős-Rényi graph with probabilities $p_{ij} = \langle x_i, x_j \rangle$, where $x_i \in \mathbf{R}^d$ is the latent position of node i (try a spectral embedding of the adjacency or Laplacian matrix).

ptype: probabilistic type inference **T. Ceritli et al. (2019)**

Infer data types in CSV files with probabilistic grammars or probabilistic finite state machines (one per type, allowing for missing values, disguised missing values (e.g., -1 or 999) and errors), with their transition probabilities estimated from data.

WarpedGANSpace: finding non-linear RBF paths in GAN latent space **C. Tzelepis et al.**

To find interpretable paths in the latent space of a GAN, without supervision, use the gradient of several sums of RBFs, trained to be easily distinguishable by a discriminator (a non-linear generalization of PCA).

Do vision transformers see like convolutional neural networks? **M. Raghu et al. (2021)**

To compare the information contained in two layers (of the same network, or not), compute the Gram matrices of the activations, $K = XX'$, $L = YY'$ (they measure the similarity of a pair of data points according to the layer) and compute the *centered kernel alignment* (CKA)

$$\text{CKA}(K, L) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K) \text{HSIC}(L, L)}},$$

where HSIC is the scalar product of the centered Gram matrices

$$\begin{aligned} \text{HSIC}(K, L) &= \text{vec } \tilde{K} \cdot \text{vec } \tilde{L} / (m - 1)^2 \\ \tilde{K} &= HKH \\ \tilde{L} &= HLH \\ H &= I + n - \frac{1}{n} \mathbf{1}\mathbf{1}' \end{aligned}$$

Compare the first and last layer of a ViT or of a CNN, the subset of activations of heads focusing on local

(resp. global) information (defined by the average distance between the query patch and the patch attended to): ViTs have more comparable layers.

To measure the importance of skip connections $z_i \mapsto z_i + f(z_i)$, look at $\|z_i\| / \|f(z_i)\|$: ViTs propagate more information through skip connections than CNNs.

Evolution through large models
J. Lehman et al. (2022)

Use a large language model, not to generate programs, but diffs; use those diffs as mutation operators, for genetic programming.

Training data-efficient image transformers and distillation through attention
H. Touvron et al.

More efficient ViT.

Hyperbolic neural networks++
R. Shimizu et al.

Split, concatenation, convolution, attention can be generalized to hyperbolic networks.

Graph-regularized tensor regression: a domain-aware framework for interpretable multi-way financial modelling
Y.L. Xu et al.

Tensor regression is a generalization of linear regression $y = \langle W, X \rangle + b$, for instance

$$y^t = w_{ij\tau} x_{ij\tau}^t + b$$

i : stock
 j : feature
 τ : lag
 t : time

where $x_{ij\tau}^t$ is the value of feature j for stock i at time $t - \tau$. We can make the model more parsimonious with a CP decomposition

$$w_{ij\tau} = \sum_c u_{ic}^1 u_{jc}^2 u_{\tau c}^3.$$

Given a graph on one of the dimensions, e.g., sector membership for the stocks (or a chain graph $\bullet \cdots \bullet$ for the lags, or some a priori relation between the features), with Laplacian L , one can add a penalty $\text{tr}(u_i^{1'} L u_i^1)$ to make the signal smooth.

Quasi-Monte Carlo methods for computing derivatives sensitivities on the GPU
P. Bilokon et al. (2022)

The *pathwise method* computes

$$\frac{\partial}{\partial S_0} \mathbb{E}_{S \sim \text{rw}(\mu, \sigma^2)} [f(S_T)],$$

with Monte Carlo simulations, as

$$\mathbb{E} \left[\frac{\partial f(S_T)}{\partial S_T} \frac{\partial S_T}{\partial S_0} \right]$$

where $\frac{\partial S_T}{\partial S_0} = \frac{S_T}{S_0}$.

Multilevel Monte Carlo and its applications in financial engineering
D. Sinha and S.P. Chakrabarty

Multilevel Monte Carlo (MLMC) uses different resolutions as control variates

$$\mathbb{E}[Y_3] = \mathbb{E}[Y_1] + \mathbb{E}[Y_2 - Y_1] + \mathbb{E}[Y_3 - Y_2]$$

where Y_1 and Y_2 are coarse, fast-to-compute approximations of Y_3 .

Combine with importance sampling.

The pursuit of balance: an overview of covariate-adaptive randomization techniques in clinical trials
Y. Lin et al. (2015)

Simple randomization can lead to accidental bias. Try:

- Block randomization;
- Stratified randomization
- Minimization: allocate the new subject (with high probability, e.g., $p = 0.8$) to produce the smallest total imbalance (variance, standard deviation, range, etc.) over its baseline characteristics;
- Dynamic hierarchical randomization: idem, after putting the baseline characteristics in a tree.

Modifications to a classic BFGS library for use with SIMD-equipped hardware and an AAD library
E. Goncharov and A. Rodrigues

Even if the objective is not parallelizable, some steps in the algorithm are:

- Line search (naive, or with a polynomial regression);
- Partial derivatives.

Liquidity stress testing in asset management
T. Roncalli et al. (2020)

Stress testing includes:

- Redemptions (liability liquidity);
- Market impact (asset liquidity);
- The relation between the two (asset-liability liquidity matching).

Machine learning time series regression with an application to nowcasting
A. Babii et al. (2021)

MIDAS with a sparse group lasso penalty

$$\alpha \|\beta\|_1 + (1 - \alpha) \|\beta\|_{2,1}$$

(i.e., lasso and group lasso), promoting sparsity between and within groups.

Moment sum-of-squares approach for fast risk estimation in uncertain environments

A.M. Jasour et al.

Given $x \in \mathbf{R}^n$ (e.g., robot position), we want to know if $x \in \chi$, where $\chi \subset \mathbf{R}^n$ is a semi-algebraic set. Both x and χ are only known approximately, but we know the moments of their distributions (for χ , the moments of the distributions of the coefficients of the polynomials defining it). The resulting semi-definite program becomes intractable as the size of the problem increases, but one can efficiently compute bounds of the moments.

Randomized geometric tools for anomaly detection in stock markets

C. Bachelard et al.

Uniformly sampling long-only fully-invested portfolios with a prescribed volatility is equivalent to sampling from $K = \mathbf{S}^{n-1} \cap \Delta$ for some simplex Δ . It is not connected, but one can:

- Identify the connected components by looking at how the 1-simplex Δ intersects the sphere \mathbf{S}^{n-1} ;
- Compute the volume of each component.

We can sample, uniformly, from a component K_i , with Monte Carlo, starting from a point p :

- Great cycle walk:
 - Pick a geodesic (great circle) ℓ going through p ;
 - Compute the intersection $\ell \cap K_i$;
 - Pick a point at random on $\ell \cap K_i$;
- Reflective great circle walk:
 - Pick a geodesic;
 - Pick a length, $L = -\tau \log \eta$, $\eta \sim U(0, 1)$;
 - Follow the geodesic for a length L , reflecting on $\partial\Delta$ whenever we reach it.

Interpretable selective learning in credit risk

D. Chen et al.

Fit a logistic regression (for credit scoring); train a neural network for the same task; train another neural network to predict when the neural net is correct and the logistic regression is not [that is a mixture of experts].

Neural variance reduction for stochastic differential equations
P.D. Hinds and M.V. Tretyakov (2022)

Use a neural network to compute control variates to reduce variance when solving SDEs (i.e., computing $E_X[f(X_T)]$).

New financial ratios based on the compositional data methodology
S. Linares-Mustarós et al.

Replace financial ratios of the form a/b , $a/(b+c)$, etc., which are asymmetric, with $\log(a/b)$, $\log(a/\sqrt{bc})$, etc.

Monte Carlo estimation of CoVaR

W. Huang et al.

The CoVaR is defined by

$$P[Y \leq \text{CoVaR}_{\alpha, \beta} \mid X = \text{VaR}_{\alpha} X] = \beta.$$

Its naive Monte Carlo estimation converges in $O(n^{-1/3})$ (we are conditioning on $X = \text{VaR}_{\alpha} X$, which has zero probability, and not on $X \leq \text{VaR}_{\alpha} X$), but it can be brought down to $O(n^{-1/2})$, with importance sampling and a Taylor expansion.

A multimodal embedding-based approach to industry classification in financial markets

R. Dolphin et al. (2022)

Word2Vec-like model, to compute company embeddings, by modeling $P[\text{target}|\text{context}]$, where the context comes from past prices and financial news; turn into a graph by adding an edge whenever cosine similarity is above 0.6.

FinRL-Meta: market environments and benchmarks for data-driven financial reinforcement learning

X.Y. Liu et al. (2022)

Gym-compatible financial market environments for reinforcement learning.

Applying separative non-negative matrix factorization to extra-financial data

P. Fogel et al. (2022)

Nonnegative matrix factorization (NMF) is asymmetric: it pays more attention to high values than low values. Instead, decompose the data as $X = X_0 + X_+ - X_-$, $X_+ \geq 0$, $X_- \geq 0$, for some baseline X_0 (e.g., the column-wise medians) and use a nonnegative tensor factorization for the order-3 tensor $[X_+ | X_-]$.

Python implementation in `nmtf`; application to ESG scores.

Deep differentiable reinforcement learning and optimal trading

T. Jaisson (2022)

Differentiable reinforcement learning (RL), for a single asset ($n = 1$), on simulated data with a 2-scale alpha

$$\begin{aligned} \alpha_t &= \alpha_t^{\text{slow}} + \alpha_t^{\text{fast}} \\ \alpha_t^{\text{slow}} &= \rho^{\text{slow}} \alpha_{t-1}^{\text{slow}} + \text{noise} \\ \alpha_t^{\text{fast}} &= \rho^{\text{fast}} \alpha_{t-1}^{\text{fast}} + \text{noise} \\ \rho^{\text{fast}} &= 0, \quad \rho^{\text{slow}} > 0 \end{aligned}$$

***Distributionally robust
end-to-end portfolio construction***
G. Costa and G.NB. Iyengar (2022)

In the portfolio optimization

$$\begin{aligned} & \text{Minimize Risk}(w) - \gamma w'y, \\ & \quad w \geq 0 \\ & \quad w'1 = 1 \end{aligned}$$

use, as risk, the worst *deviation risk measure*

$$\begin{aligned} \text{Risk}(w) &= \max_{\substack{p \geq 0 \\ p'1=1 \\ D_\phi(p, \text{Unif}) \leq \delta}} f_\varepsilon(w, p) \\ f_\varepsilon(w, p) &= \min_c \sum_j p_j R(w'\varepsilon_j - c) \\ D_\phi: & \phi\text{-divergence} \\ \varepsilon: & \text{past prediction errors} \\ R: & \text{even function} \end{aligned}$$

The minimax problem can be reformulated with convex duality, and solved end-to-end (to maximize the Sharpe ratio), for 20 assets, 15 years, weekly returns, 17 predictors (code available).

Supervised portfolios
G. Chevalier et al. (2022)

Instead of forecasting returns (or uniformized returns, residual returns, volatility-adjusted returns, etc.) forecast the weights of the optimal portfolio (using realized returns as alpha). This adjusts for: factor exposures, volatility, investor preferences (utility, constraints). Example with 25 assets, 14 features.

Alternatives to deep neural networks in finance
A.V. Antonov and V.V. Piterbarg (2022)

To approximate functions $f: \mathbf{R}^k \rightarrow \mathbf{R}$, try

$$\tilde{f}(x) = \sum \alpha_n s(x - z_n).$$

For the Fourier transform, the z_n 's are on a grid, and $\alpha_n = \hat{f}(z_n)$ – but we could use quasi-random points for the z_n 's, estimate the α_n 's with a linear regression, and fine-tune the position of the z_n 's (*stochastic sampling*).

For *image rendering*, s is sinc or its Lanczos generalization

$$s(x) = \frac{\sin(\pi x)}{\pi x} \quad \text{or} \quad s(x) = \frac{\sin(\pi x)}{\pi x} \frac{\sin(\pi x/a)}{\pi x/a} \mathbf{1}_{x \in [-a, a]}.$$

Generalized stochastic sampling uses

$$\tilde{f}(x) = \sum_n \alpha_n \prod_d \phi((x_d - z_{nd})\beta_d)$$

for a fixed activation function ϕ (e.g., sinc), quasi-random z_n 's, and

- $\beta_d = \lambda_d \cdot \eta$, $\lambda_d^2 = \mathbf{E}(\nabla_d f(x))^2$, η optimized;
- or β_d optimized;
- or $\beta_{d,n}$ optimized.

The tensor train decomposition

$$\begin{aligned} A_{i_1, \dots, i_D} &= G_{i_1 k_1}^1 G_{k_1 i_2 k_2}^2 \cdots G_{k_{D-2} i_{D-1} k_{D-1}}^{D-1} G_{k_{D-1} i_D}^D \\ A(i_1, \dots, i_D) &= \underbrace{g_1(i_1)}_{\text{row vector}} \underbrace{g_2(i_2) \cdots g_{D-1}(i_{D-1})}_{\text{matrices}} \underbrace{g_D(i_D)}_{\text{column vector}} \end{aligned}$$

can be made functional (fTT)

$$\tilde{f}(x) = \underbrace{g_1(x_1)}_{\text{row vector}} \underbrace{g_2(x_2) \cdots g_{D-1}(x_{D-1})}_{\text{matrices}} \underbrace{g_D(x_D)}_{\text{column vector}}$$

where the g_d 's are learned (with alternating least squares, from basis functions – note that this depends on the order of the x_d 's).

Classical approaches include:

- Fourier decomposition;
- Chebychev decomposition (up to dimension 7: $7^7 < 10^6$);
- Natural neighbour interpolation (in dimension 2 or 3) or other linear interpolations (barycenters).

***Cryptocurrency bubble detection:
a new stock market dataset, financial task
and hyperbolic models***
R. Sawhney et al.

Identify bubbles in cryptocurrencies (9 exchanges \times 50 currencies, 5 years of daily OHLC prices) with the PSY model, and train a hyperbolic GRU on both text (tweets with currency tickers, e.g., \$DOGE) and prices, to predict the probability that a bubble starts or ends at time t .

In the Poincaré ball:

$$\begin{aligned} g_x &= \frac{2}{1 - \|x\|^2} g_{\text{Euclidean}} \\ x \oplus y &= \frac{(1 + 2\langle x, y \rangle + \|y\|^2)x + (1 - \|x\|^2)y}{1 + 2\langle x, y \rangle + \|x\|^2 \|y\|^2} \\ \exp_x(v) &= x \oplus \left(\tanh\left(\frac{\|v\|}{1 - \|x\|^2}\right) \frac{v}{\|v\|} \right) \\ \log_x(y) &= (1 - \|x\|^2) \tanh^{-1}(\| -x \oplus y \|) \frac{-x \oplus y}{\| -x \oplus y \|} \\ W \otimes x &= \exp_0(W \log_0(x)) \end{aligned}$$

***Testing for multiple bubbles:
historical episodes of exuberance
and collapse in the S&P 500***
P.C.B. Phillips et al. (2015)

To detect a (single) price bubble, try the following test statistics:

$$\begin{aligned} \text{PYW:} \quad \text{SADF}(r_0) &= \sup_{r_0 \leq r_2 \leq 1} \text{ADF}_{[0, r_2]} \\ \text{PSY:} \quad \text{GSADF}(r_0) &= \sup_{\substack{r_0 \leq r_2 \leq 1 \\ 0 \leq r_1 \leq r_2 - r_0}} \text{ADF}_{[r_1, r_2]}. \end{aligned}$$

They can be adapted to the detection of multiple bubbles.

Deep multiple instance learning for forecasting stock trends using financial news
Y. Deng and S.M. Yiu

In *multiple instance learning*, training instances are arranged in bags, and only bags are labeled, not instances.

Forecast the sign of the next day's returns from news:

- Word embedding, pretrained with Glove;
- BiLSTM;
- Instance-level classifiers;
- Aggregation;
- Final classifier.

The RIFT (representation of inter-related time series) model and its applications
A. Sokolov et al. (2022)

Compute a neural representation of financial time series by applying

- A TCN on stock returns;
- A transformer on industry and market returns;
- An industry embedding

in a Siamese network to forecast future correlations. [Instead, forecast reversion or divergence of pairs, or cointegration, or mutual information.]

Decomposing cross-sectional volatility
J. Menchero and A. Morozov (2010)

Given a risk model

$$X_i = \sum_k \beta_{ik} F_k + u_i,$$

the cross-sectional volatility can be decomposed as

$$\sigma(X_i) = \sum_k F_k \sigma(\beta_{ik}) \rho(\beta_{ik}, X_i) + \sigma(u_i) \rho(u_i, X_i).$$

AdaptSPEC: adaptive spectral estimation for nonstationary time series
O. Rosen et al. (2012)

Model (oscillating) non-stationary time series by recursively splitting them and estimating their spectrum on each segment, with reversible jump Markov chain Monte Carlo (RJMCMC).

Statistical indetence of lead-lag at various timescales between asynchronous time series from p-values of transfer entropy
C. Bongiorno and D. Challet (2022)

Asymptotic distribution of the test statistic

$$H_0 : \text{TE}(B \rightarrow A) = \text{TE}(C \rightarrow B)$$

$$H_1 : \text{TE}(B \rightarrow A) > \text{TE}(C \rightarrow B)$$

where the transfer entropy

$$\text{TE}(B \rightarrow A) = H(A^+|A) - H(A^+|A, B)$$

(A^+ denoted the future of A) is a generalization of Granger causality, often approximated by discretizing the data. (If you are patient, you can also compute bootstrap p -values.)

Advances in domain independent linear text segmentation
F.Y.Y. Choi

To segment a text:

- Split it into sentences;
- Remove stopwords, step the words, count the words;
- Compute the matrix of similarities between sentences;
- Replace each value with its rank in a local region;
- Compute s_{ij} , the sum of the values of $\text{sim}_{[i,j] \times [i,j]}$ (start along the diagonal and move outwards);
- Recursively split $[1, N]$, maximizing the density $\sum s_k / \sum a_k$, where a_k is the area of $k \times k$, $k = [i, j]$;
- To decide when to stop, compute the differences $\delta D_n = D_n - D_{n-1}$, smooth them, compute their mean μ and standard deviation σ ; stop when $\delta D > \mu + 1.2\sigma$.

Self-attention between datapoints: going beyond individual input-output pairs in deep learning
J. Kossen et al. (2021)

Do not use attention only between attributes, but also between datapoints (samples – this assumes you put the whole dataset in each minibatch or, at least, a representative part of it). Try on tabular data.

Decision transformer: reinforcement learning via sequence modeling
L. Chen et al.

To learn, offline, from suboptimal trajectories, train a GPT model to predict the next token in a sequence of return-to-go, state, action. For instance, one can find shortest paths on a graph by training on random walks.

```
t = embed_t(t) # Positional embedding
s = embed_s(s)
a = embed_a(a)
R = embed_R(R)
return transformer( s+t, a+t, R+t ).action
```

CrossViT: cross-attention multi-scale vision transformer for image classification
C.F. Chen et al.

Process both small and large image patches with a ViT; to combine them, use *cross-attention*, i.e., attention between the CLS token for one patch size and the image patch tokens for the other size – the CLS tokens can be seen as “inducing points” (or tokens): the computations only require linear time.

Efficient and modular implicit differentiation

M. Blondel et al. (2022)

Implicit differentiation (computing the gradient of the solution of an optimization problem) in JAX (for optimization layers, or bilevel problems: hyperparameter optimization, meta-learning): separately specify the optimality conditions and implement the optimization.

Evaluating robustness of neural networks with mixed integer programming

V. Tjeng et al. (2019)

Neural networks can be verified (their robustness to adversarial examples can be measured) efficiently, up to 100,000 ReLU, with mixed integer programming (MIPVerify.jp).

Random dot product graph models for social networks

S.J. Young and E.R. Scheinerman

The dot product random graph model generates graphs by sampling a vector X_u (from some probability distribution) and adding an edge $u-v$ with probability $\langle X_u, X_v \rangle$. For directed graphs, use two distributions $p(u-v) = \langle X_u, Y_v \rangle$. Other random graph models include:

- Configuration model (prescribed degree distribution);
- Preferred attachment (Barabási-Albert);
- Copying model.

Duplication models for biological networks

F. Chung et al. (2003)

The degree distribution of biological graphs follows a power law with exponent $\beta \in (1, 2)$, in contrast with non-biological networks, for which $\beta \in (2, 4)$. This can be modeled by a duplication process:

- Start with a graph G_0 ;
- Pick a node at random;
- Duplicate it, keeping each edge with probability p ;
- Iterate.

Asymptotically, the power law exponent β satisfies $p(\beta - 1) = 1 - p^{\beta-1}$; in particular, if $p \in (\frac{1}{2}, 1)$, then $\beta < 2$.

Modernizing PHCpack through phcpy

J. Verschelde (2014)

PHCpack is an (old, file and menu-based) package to solve systems of polynomial equations (over \mathbf{C}^n) $f(x) = 0$ using a system with known solutions $g(x) = 0$ by keeping track of the solutions of $h_t(x) = 0$ where

$$h_t(x) = \gamma(1-t)g(x) + tf(x), \quad t \in (0, 1)$$

and $\gamma \in \mathbf{C}$ is random.

Multivariate backtests and copulas for risk evaluation

B. David and G. Zumbach (2022)

Fit a bivariate Student copula to LM-GARCH innovations (to remove heteroskedasticity), in-sample; then apply the *Rosenblatt transform* to out-of-sample

$$\begin{cases} \text{Student copula} & \longrightarrow & \text{Uniform copula} \\ (u_1, u_2) & \longmapsto & (u_1, P[U_2 \leq u_2 \mid U_1 = u_1]) \end{cases}$$

and test if the result is indeed uniform.

Polynomial voting rules

W. Tang and D.D. Yao (2022)

In a proof of stake (PoS) system, agents have a voting power proportional to

- Their stake;
- The square root of their stake;
- Their randomly fluctuating stake: each bidder receives a new stake with probability proportional to some power of its current stake.

On finding the community with maximum persistence probability

A. Avellone et al.

The persistence probability of a subgraph $C \subset V$ of a graph $G = (V, E)$ is

$$\alpha = \frac{\sum_{i,j \in C} a_{ij}}{\sum_{i \in C, j \in V} a_{ij}}.$$

For small graphs, it can be computed with a mixed integer program (MIP); for larger graphs, heuristics are available.

Compromise-free Bayesian neural networks

K. Javid et al.

The *Bayesian evidence* (aka *marginal likelihood*) is the average of the likelihood function over the parameter space, weighted by the prior distribution. It is often a proxy for the out-of-sample performance.

Randomized Nyström preconditioning

Z. Frangella et al.

To solve $(A + \mu I)x = b$, with A positive semi-definite, $\mu \geq 0$, consider a randomized Nyström approximation

$$\Omega \sim N(0_{n \times \ell} I)$$

$$\hat{A} = (A\Omega)(\Omega' A \Omega)^\dagger (A\Omega)',$$

compute its eigendecomposition $\hat{A} = U\Lambda U'$ and use the preconditioner

$$P = \frac{U(\Lambda + \mu I)U'}{\lambda_\ell + \mu} + (I - UU')$$

i.e., replace $(A + \mu I)$ with $P^{-1}(A + \mu I)$ (it is easy to solve $Py = 0$).

Binarsity: a penalization for one-hot encoded features in linear supervised learning
M.Z. Alaya et al. (2019)

Discretize (binarize) continuous variables into b bins and fit a linear model with a fused lasso (total variation) penalty to make the transformation locally constant, and a constraint to have the sum of the weights sum to zero for each predictor.

To total variation proximal operator can be computed efficiently, and applied separately from the constraint.

Quantifying the impact of ecological memory on the dynamics of interacting communities
M. Khalighi et al. (2021)

To add memory to the Lotka-Volterra model, replace d/dt with the fractional derivative \mathfrak{D}^μ , where $\mu \in (0, 1)$ measures the memory.

$$\mathfrak{D}^\mu g(t) = \frac{1}{\Gamma(1-\mu)} \int_{t_0}^t \frac{g'(\tau) d\tau}{(t-\tau)^\mu}$$

Three-species Lotka-Volterra model with respect to Caputo and Caputo-Fabrizio fractional operators
M. Khalighi et al. (2021)

The fractional derivative is not unique; popular definitions include (Caputo, Caputo-Fabrizio)

$$\begin{aligned}\mathfrak{D}^\alpha f(t) &= \frac{1}{\Gamma(1-\alpha)} \int_0^t \frac{f'(\tau) d\tau}{(t-\tau)^\alpha} \\ \mathfrak{D}^\alpha f(t) &= \frac{1}{1-\alpha} \int_0^t \exp\left[-\frac{\alpha(t-\tau)}{1-\alpha}\right] f'(\tau) d\tau.\end{aligned}$$

Conic optimization via operator splitting and homogeneous self-dual embedding
B. O'Donoghue et al. (2016)

The pair of primal and dual problems

$$\begin{array}{ll}\text{Find} & x, s \\ \text{To minimize} & c'x \\ \text{Such that} & Ax + s = b \\ & x \in \mathbf{R}^n \\ & s \in K\end{array} \quad \begin{array}{ll}\text{Find} & y, r \\ \text{To maximize} & -b'y \\ \text{Such that} & -A'y + r = c \\ & r = 0 \\ & y \in K^*\end{array}$$

can be converted into a feasibility problem

$$\begin{pmatrix} r \\ s \\ \kappa \end{pmatrix} = \begin{pmatrix} 0 & A' & c \\ -A & 0 & b \\ c' & b' & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ \tau \end{pmatrix} \quad \begin{array}{l} \text{dual constraint} \\ \text{primal constraint} \\ \text{duality gap} \end{array}$$

or, after introducing scaling factors $\tau, \kappa \geq 0$ to detect primal or dual infeasibility (*homogeneous self-dual embedding*)

$$\begin{pmatrix} r \\ s \\ \kappa \end{pmatrix} = \begin{pmatrix} 0 & A' & c \\ -A & 0 & b \\ -c' & -b' & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ \tau \end{pmatrix}.$$

This problem is self-dual:

$$\begin{array}{ll}\text{Find} & u, v \\ \text{Such that} & v = Qu \\ & (u, v) \in C \times C^*.\end{array}$$

It can be solved with the ADMM algorithm.

$$\begin{array}{ll}\text{Find} & u, v, \tilde{u}, \tilde{v} \\ \text{To minimize} & I_{C \times C^*}(u, v) + I_{Q_{u=v}}(\tilde{u}, \tilde{v}) \\ \text{Such that} & (u, v) = (\tilde{u}, \tilde{v})\end{array}$$

An agent-based model with realistic financial time series: a method for agent-based models validation
L.G. de Faria

Long list of stylized facts, for log-returns

- Fat tails, which disappear at lower frequencies;
- Heavy tails (tail index), even after correction for volatility (GARCH);
- Equity premium: $E[r] > E[r_{\text{rf}}]$;
- Excess volatility: $\sigma(\text{returns}) > \sigma(\text{fundamentals})$;
- Leverage: $\text{Cor}(r_{\text{past}}, \sigma_{\text{future}}) < 0$;
- No autocorrelation;
- Long memory: $\text{Cor}(|r_t|, |r_s|)$;
- Power law of returns ("inverse cubic law") and of volatility;
- Volatility clustering: $\text{Cor}(\sigma_t, \sigma_{t+1})$;
- $\text{Cor}(\sigma, \text{volume}) > 0$

volume:

- Power law;
- Long memory

inter-trade duration:

- Clustering;
- Long memory;
- Over-dispersion

transaction size:

- Power law

spreads:

- $\text{Cor}(\text{spread}, \sigma) > 0$;
- $\text{Cor}(\text{spread}, \text{volume}) < 0$.

Improving graph neural network expressivity via subgraph isomorphism counting
G. Bouritsas et al.

Graph neural nets (GNN) are blind to structural properties, such as triangles and larger cycles. Add node and edge features counting small subgraphs containing a given node or edge.



***Weisfeiler and Leman go sparse:
towards scalable higher-order graph embeddings***
C. Morris et al. (2020)

Generalize the Weisfeiler-Lehman (WL) graph isomorphism algorithm by considering k -tuples of nodes, two tuples being neighbours if they differ by only one node, and if those nodes are neighbours.

***A theoretical comparison
of graph neural network extensions***
P.A. Papp and R. Wattenhofer (2022)

To increase the expressiveness of GNNs:

- Add node features, e.g., the number of cycles of length k (or some other motif) containing that node;
- Add, as node features, the isomorphism class of the k -hop neighbourhoods;
- Drop k nodes at random, and consider the resulting collection of graphs;
- Mark k nodes at random, and consider the resulting collection of graphs.

***SpeqNets: sparsity-aware
permutation-equivariant graph networks***
C. Morris et al. (2022)

The k -dimensional Weisfeiler-Lehman algorithm (k -WL) generalizes the WL algorithm (1-WL) by considering all k -tuples of nodes (there are exponentially many). The (k, s) -LWL algorithm considers a subset of all k -tuples, viz those whose induced graph has at most s components.

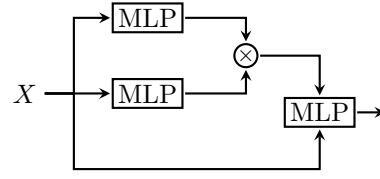
***Spectre: spectral conditioning
helps to overcome the expressivity limits
of one-shot graph generators***
K. Martinkus et al. (2022)

Generate graphs, with a GAN, by conditioning on the first eigenvalues and eigenvectors of the Laplacian matrix (which encode global properties):

- First, generate eigenvalues;
- Then, generate the eigenvectors, conditioned on the eigenvalues, starting with a bank of orthogonal matrices, and multiplying them, on both sides, by orthogonal matrices (exponentials of skew-symmetric matrices, computed by a PointNetST); note that not all eigenvalues and eigenvectors come from valid Laplacians;
- Use a PPGN (provably powerful graph network) to refine the approximate Laplacian, $L = U \text{diag}(\Lambda) U'$, and convert it to an adjacency matrix;
- Use a discriminator for each step; the last one ensures the adjacency matrix is consistent with the eigenvalues and eigenvectors.

Provably powerful graph networks
H. Maron et al. (2019)

Adding a matrix multiplication layer to GNNs increases their expressiveness to 3-WL.



$$X \in \mathbf{R}^{h \times n \times n}$$

$$M_1 = \text{MLP}_1(X) \in \mathbf{R}^{h \times n \times n}$$

$$M_2 = \text{MLP}_2(X)$$

$$M^i = M_1^i M_2^i \in \mathbf{R}^{n \times n}$$

$$M \in \mathbf{R}^{n \times n}$$

$$Y = \text{MLP}_3(X \parallel M) \in \mathbf{R}^{h \times n \times n}$$

repeat n times, with skip-connections

Local augmentation for graph neural networks
S. Liu et al. (2022)

To augment data, find similar nodes and look at their neighbourhoods.

***Janossy pooling:
learning deep permutation-invariant
functions for variable-size inputs***
R.L. Murphy et al. (2019)

Arbitrary permutation-invariant functions can be defined as averages (of permutation-sensitive functions f) over all orderings.

$$\text{Mean}_{\sigma \in \mathfrak{S}_n} f(x_\sigma)$$

They can be approximated using:

- Canonical orderings;
- Functions f using only their first k arguments, $f(x_1, \dots, x_n) = f(x_1, \dots, x_k)$, $k \ll n$, i.e., accounting for order- k interactions at most;
- Random orderings (and stochastic optimization).

The permutation-sensitive function f could be an LSTM.

***G-Mixup: graph data augmentation
for graph classification***
X. Han et al.

To apply Mixup to graphs, estimate a graphon for each class, and sample from a convex combination of them.

To estimate a graph from a set of graphs, first align them by sorting their nodes by degree, estimate a step function graphon for each graph, and average them – this assumes the marginal $\int W(x, y) dx = W(y)$ is very different from a constant function.

**GenLabel: Mixup relabeling
using generative models**
J.Y. Sohn et al. (2022)

Mixup can suffer from *manifold intrusion*: mixing two classes may intrude into the manifold of another one.



Additionally, the linear labeling is suboptimal for softmax regression. Replacing

$$y_{\text{mix}} = \lambda e_1 + (1 - \lambda) e_2$$

with

$$y_{\text{gen}} \propto \sum_i \hat{p}_i(x_{\text{mix}}) e_i,$$

where $\hat{p} = p(x|y)$ is a generative model, addresses both issues.

**Large-scale representation learning on graph
via bootstrapping**
S. Thakoor et al.

For self-supervised training on graph data, consider two augmentations, g_1 , g_2 of a graph, and train two encoders, e_1 such that $e_1(g_1) \approx e_2(g_2)$, and set $e_2 = \text{EMA}(e_1)$.

**Structure-aware transformer
for graph representation learning**
D. Chen et al. (2022)

Transformers augment the data with a positional embedding. For graphs, node distances are not enough: also add structural information.

**Interpretable and generalizable graph learning
via stochastic attention mechanism**
S. Miao et al. (2022)

Add noise *inside* the network (in the attention mechanism); this also helps identify task-relevant subgraphs.

**Mention memory: incorporating textual
knowledge into transformers
through entity mention attention**
M. de Jong (2022)

Combine language model and knowledge base (entity mention embeddings) with an attention mechanism.

**Logical rule induction and theory learning
using neural theorem proving**
A. Campero et al.

Use a dense representation of facts as (V, S, O, belief) and rules as

$$\begin{matrix} (V_0 S_0 O_0 & V_1 S_1 O_1 & V_2 S_2 O_2) \\ \text{conclusion} & \text{premise} & \end{matrix}$$

and define the belief of a conclusion as

$$\langle v, V_0 \rangle \langle s, S_0 \rangle \langle o, O_0 \rangle \langle v_1, V_1 \rangle \langle s_1, S_1 \rangle \langle o_1, O_1 \rangle \langle v_2, V_2 \rangle \langle s_2, S_2 \rangle \langle o_2, O_2 \rangle$$

and use forward chaining for derive new facts.

**Learning hierarchy-aware knowledge graph
embeddings for link prediction**
Z. Zhang et al. (2020)

Use polar coordinates to find an embedding of entities (head, tail) and relations accounting for hierarchy.

$$\text{score} = -\|h_m \odot r_m - t_m\|_2 - \lambda \|\sin(h_p + r_p - t_p)\|_1$$

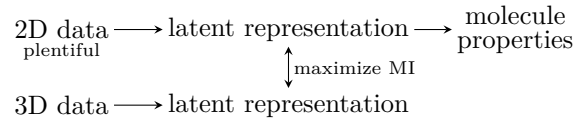
There are many other score functions: TransE, RotatE, ComplEx, etc.

**Translating embeddings
for modeling multi-relational data**
A. Bordes et al.

TransE embeds entities as vectors and relations as translations (as in the semantic interpretation of word2vec: king-man+woman).

**RotatE: knowledge graph embedding
by relational rotation in complex space**
Z. Sun et al.

**3D Infomax improves GNNs
for molecular property prediction**
H. Stärk et al. (2022)



E(n) equivariant graph neural network
V.G. Satorras et al. (2021)

Replace the GNN layer with

$$m_{ij} = \phi_2(h_i^\ell, h_j^\ell, a_{ij})$$

$$m_i = \sum_{j \in N(i)} m_{ij}$$

$$h_i^{\ell+1} = \phi_h(h_i^\ell, m_i)$$

with

$$m_{ij} = \phi_2(h_i^\ell, h_j^\ell, \|x_i^\ell - x_j^\ell\|^2, a_{ij})$$

$$x_i^{\ell+1} = x_i^\ell + \text{Mean}_{j \neq i}(x_i^\ell - x_j^\ell) \phi_x(m_{ij})$$

$$m_i = \text{unchanged}$$

$$h_i^{\ell+1} = \text{unchanged}$$

**Vector neurons: a general framework
for $SO(3)$ -equivariant networks**
C. Deng et al.

Replace 1-dimensional (scalar) neurons with 3-dimensional ones: linear transformations

$$f_W : \begin{cases} \mathbf{R}^{c \times 3} \longrightarrow \mathbf{R}^{c' \times 3} \\ V \longmapsto WV \end{cases}$$

are naturally equivariant, $f_W(V)R = (WV)R = W(VR) = f_W(VR)$, but the non-linearities have to be changed, e.g., to $V \mapsto V'$, where

$$\begin{aligned} V &= (v_1, \dots, v_c) \in \mathbf{R}^{c \times 3} \\ W_c, U_c &\in \mathbf{R}^{1 \times C} \text{ for each } c \in \llbracket 1, C \rrbracket \\ q_c &= W_c V, \quad k_c = U_c V \in \mathbf{R}^{1 \times 3} \\ v' &= \begin{cases} q_c & \text{if } \langle q_c, k_c \rangle = 0 \\ q_c - \left\langle q_c, \frac{k_c}{\|k_c\|} \right\rangle \frac{k_c}{\|k_c\|} & \text{otherwise.} \end{cases} \end{aligned}$$

(The normalization layers are easy to make equivariant.)

**On the equivalence between temporal and static
equivariant graph representations**
J. Gao et B. Ribeiro (2022)

Instead of time-and-graph representations (compute node embeddings for each graph G_t in the sequence, and then model the evolution of those embeddings), try time-then-graph: compute representations of the time series of node features (e.g., with an LSTM), and then embed them.

**Inductive representation learning
on temporal graphs**
D. Xu et al. (2020)

Dynamic graphs can be seen as graphs with another type of edge, for the time evolution; temporal graph attention (TGAT) layers use the attention mechanism in the time dimension (in the space dimension, use a GCN, GAT, etc.)

**Temporal graph networks
for deep learning on dynamic graphs**
E. Rossi et al.

To process dynamic graphs (in discrete or continuous time, *i.e.*, sequences of graphs, or time-stamped lists of graph events), replace the node (or edge) features with time series (LSTM, attention, etc.).

**Cycle representation learning
for inductive relation prediction**
Z. Yan et al. (2022)

Given a graph (V, E) , its incidence matrix $\partial \in \mathbf{F}_2^{|V| \times |E|}$ defines a map $\partial : \mathbf{F}_2^{|E|} \rightarrow \mathbf{F}_2^{|V|}$; its kernel is the set of cycles, Z .

To get a basis of Z , pick a node p and build its shortest path tree T_p : the edges not in T_p define cycles, which form a basis of Z , with $\beta_1 = |E| - |V| + 1$ elements.

(To have shorter cycles, repeat for several points, e.g., the cluster centers from spectral clustering.)

The cycle incidence matrix of this basis is $C_{T_p} \in \mathbf{F}_2^{|E| \times \beta_1}$.

Use a bidirectional LSTM to compute cycle features.

Build a new graph, with cycles as nodes, and edge weights equal to the number of edges the cycles have in common.

To test if an edge should be in the graph, add it to the graph, and use a GNN on the cycle graph to compute its “confidence”.

**GNNRank: learning global rankings
from pairwise comparisons
via directed graph neural networks**
Y. He et al. (2022)

Given a set of pairwise comparisons between n elements as the (weighted) adjacency matrix A of a directed graph, *serial rank* computes the binary comparison matrix $C_{ij} = \text{sign}(A_{ij} - A_{ji})$, then the similarity matrix $S = \frac{1}{2}(n\mathbf{1}\mathbf{1}' + CC')$, and finally its Fiedler vector (eigenvector for the second largest eigenvalue – the first one is $\mathbf{1}$), whose coordinate define the desired order.

GNNRank replaces the computation of the similarity matrix $A \mapsto S$ with a GNN, and the computation of the Fiedler vector with a few proximal gradient steps for the constrained optimization problem

$$\begin{aligned} \text{Find} \quad & x \\ \text{To minimize} \quad & x' S x \\ \text{Such that} \quad & \|x\|_2 = 1 \\ & x' \mathbf{1} = 1 \end{aligned}$$


**G^2 CN: graph Gaussian convolution networks
with concentrated graph filters**
M. Li et al.

The action of a GNN on node features can be written as $x \mapsto g(L)x$, where L is the normalized Laplacian. Look at the maximum response, the center, and the bandwidth.

$$\begin{aligned} R &= \max_{\lambda \in [0, 2]} g(\lambda) \\ b &= \text{Argmax}_{\lambda \in [0, 2]} g(\lambda) \\ \text{bw} &= \int_0^2 \mathbf{1}(g(\lambda) \geq R/\sqrt{2}) d\lambda \end{aligned}$$

Try $g(\lambda) = e^{-T(\lambda-b)^2}$.

**Molecular representation learning
via heterogeneous motif graph neural networks**
Z. Yu and H. Gao (2022)

Molecules and motifs (, C-C, C-OH, etc.) for a bipartite graph, similar to the document-word bipartite graph in NLP: TF-IDF features can complement graph neural nets.

**Topology-aware network pruning
using multi-stage graph embedding
and reinforcement learning**
S. Yu et al. (2022)

Use reinforcement learning (RL) to find a good pruning strategy, with a reward function defined from accuracy and pruning ratio, after encoding the neural network (graph embedding) with a multi-stage GNN (to account for its hierarchical structure).

**Parametrized explainer
for graph neural network**
D. Luo et al.

GNNEExplainer gives an explanation (a subgraph) for the output of a single instance – but this has to be done anew for each sample. PGExplainer trains a GNN to output those explanations

**GNNEExplainer: generating explanations
for graph neural networks**
R. Ying et al. (2019)

To explain the output of a GNN, for a given input, look for a small subgraph, and a subset of features, such that altering those features on that subgraph significantly changes the output – mutual information $H(Y) - H(Y|S)$ measures that.

**PGMEExplainer: probabilistic graphical model
explanations for graph neural networks**
M.N. Vu and M.T. Thai (2020)

Explain the (node prediction) output of a GNN, locally, with a simple Bayesian network (built on a small subgraph (motif) containing the target node).

The CLRS algorithmic reasoning benchmark
P. Veličković et al. (2022)

Benchmarks (30 algorithms from CLRS: sort, search, dynamic programming, graphs, strings, geometry) to test whether neural networks can learn to reproduce them.

**Diffusion-LM
improves controllable text generation**
X.L. Li et al.

Diffusion models can also generate text with a prescribed semantic contents (or length, or sentence structure, etc.)

$$\begin{array}{c} X_T \rightarrow X_{T-1} \rightarrow \dots \rightarrow X_0 \rightarrow \text{text} \\ \text{noise} \qquad \qquad \qquad \text{word} \\ \qquad \qquad \qquad \qquad \qquad \text{vectors} \end{array}$$

**Photorealistic text-to-image diffusion models
with deep language understanding**
C. Saharia et al. (2022)

Contrary to Dall-E, Imagen uses a very large language model (T5-XXL), trained on text only, frozen; it fine-tunes the result with two efficient U-Net steps. It uses dynamic thresholding to avoid fully-saturated pixels.

Drawbench is a benchmark for text-to-image generation (list of prompts, challenging for various reasons).

**GLIDE: Towards photorealistic
image generation and editing
with text-guided diffusion models**
A. Nichol et al.

Diffusion models are trained by progressively adding noise to an image x_0

$$\begin{aligned} x_0 &\sim \text{Data} \\ x_t | x_{t-1} &\sim N(\sqrt{\alpha_t} x_{t-1}, (1 - \alpha_t) I) \end{aligned}$$

and trying to undo that process

$$\begin{aligned} x_T &\sim N(0, I) \\ x_{t-1} | x_t &\sim N(\mu_\theta(x_t), \Sigma_\theta(x_t)) \end{aligned}$$

(with Σ_θ diagonal). If $x_t = x_0 + \varepsilon$, the loss tries to recover ε

$$\text{Loss} = \mathbb{E}_{\substack{t \sim \text{Unif}[1, T] \\ x_0 \sim \text{Data} \\ \varepsilon \sim N(0, 1)}} \|\varepsilon - \varepsilon_\theta(x_t, t)\|^2.$$

Guided diffusion increases the likelihood of a given class y

$$\hat{\mu}_\theta(x_t | y) = \mu_\theta(x_t | y) + s \cdot \Sigma_\theta(x_t | y) \nabla_{x_t} \log p_\phi(y | x_t).$$

Classifier-free guidance also moves the model away from a “null label” \emptyset (e.g., an empty prompt)

$$\hat{\varepsilon}(x_t | y) = \varepsilon_t(x_t | \emptyset) + s(\varepsilon_\theta(x_t | y) - \varepsilon_\theta(x_t | \emptyset)).$$

CLIP-guidance uses a joint representation of images f and text g , encouraging large dot products $f(x) \cdot g(c)$ for matching pairs (contrastive cross-entropy), instead of a classifier

$$\hat{\mu}_\theta(x_t | y) = \mu_\theta(x_t | y) + s \cdot \Sigma_\theta(x_t | y) \nabla_{x_t} f(x_t) \cdot g(c).$$

Denoising diffusion probabilistic models
J. Ho et al.

Initial paper on (unconditional) diffusion models for image generation.

$$\mu_\theta = \frac{1}{\sqrt{\alpha}} \left(x - \frac{\beta}{\sqrt{1 - \alpha}} \varepsilon_\theta \right)$$

**Reverse-time stochastic differential equation
for generative modeling
L. Winkler (2021)**

Given an SDE

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW,$$

use the Kolmogorov forward equation (KFE)

$$\partial_t p(x_t) = -\partial_{x_t} [\mu(x_t)p(x_t)] + \frac{1}{2}\partial_{x_t}^2 [\sigma^2(x_t)p(x_t)]$$

and the Kolmogorov backward equation ($t \leq s$)

$$-\partial_t p(x_s|x_t) = \mu(x_t)\partial_{x_t} p(x_s|x_t) + \frac{1}{2}\sigma^2(x_t)\partial_{x_t}^2 p(x_s|x_t)$$

to derive a PDE for $p(x_s, x_t) = p(x_s|x_t)p(x_t)$:

$$\begin{aligned} -\partial_t p(x_s, x_t) = & \\ & \partial_{x_t} \left[p(x_s, x_t) \left(\mu(x_t) - \frac{\partial_{x_t} [\sigma^2(x_t)p(x_t)]}{p(x_t)} \right) \right] \\ & + \frac{1}{2}\partial_{x_t}^2 [p(x_s, x_t)\sigma^2(x_t)]. \end{aligned}$$

It is the KFE of a SDE. If the noise σ is independent of the input, we can move it outside of the partial derivative, and we recognize the log-derivative of p , the score. The final SDE is

$$d\bar{x}_t = [\mu(x_t) - \sigma_t^2 \partial_{x_t} \log p(x_t)]dt + \sigma_t d\bar{W}_t.$$

**Reverse-time diffusion equation models
B.D.O. Anderson (1980)**

**A primer on monotone operator methods
A.K. Ryu and S. Boyd (2016)**

2. A *relation* (or *operator*, or *correspondance*, or *set-valued function*) on \mathbf{R}^n is a subset $F \subset \mathbf{R}^n \times \mathbf{R}^n$. Its *subdifferential* is

$$\partial f = \{(x, g) : \forall z \ f(z) \geq f(x) + g^T(z - x)\}$$

$$\begin{aligned} (v, u) \in \partial f & \iff (u, v) \in (\partial f)^{-1} \\ & \iff v \in \underset{x}{\operatorname{Argmin}} f(x) - u^T x \\ & \iff f(v) + f^*(u) = v^T u \end{aligned}$$

where $f^*(y) = \sum_x y^T x - f(x)$.

A function $f : \mathbf{R}^n \rightarrow \mathbf{R} \cup \{\infty\}$ is *closed* if its epigraph

$$\operatorname{epi} f = \{(x, t) : x \in \operatorname{dom} f, f(x) \leq t\};$$

it is *proper* if its domain is non-empty. If f is *CCP* (convex, closed, proper), $f^{**} = f$, and $(\partial f)^{-1} = \partial f^*$.

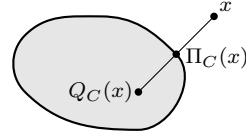
3. A Lipschitz relation with constant L is a function; if $L < 1$, it is a contraction; if $L = 1$, it is *non-expansive*. The set of fixed points of a non-expansive function, $(I - F)^{-1}(0)$, is closed and convex (but possibly empty); for a contraction, it has exactly one point.

An *averaged* operator is of the form $F = (1 - \theta)I + \theta G$, $\theta \in (0, 1)$, G non-expansive; F is still non-expansive, and has the same fixed points as G .

Projections, and *overprojections* (on a compact set C) are non-expansive.

$$\Pi_C(x) = \underset{z \in C}{\operatorname{Argmin}} \|z - x\|$$

$$Q_C = 2\Pi_C - I$$



4. A relation f is *monotone* if

$$\forall x, y \ (fx - fy)^T(x - y) \geq 0;$$

it is *maximal* if there is no larger monotone relation (for the inclusion, on $\mathbf{R}^n \times \mathbf{R}^n$); it is *strongly monotone* if $\forall x, y \ (fx - fy)^T(x - y) \geq m \|x - y\|^2$ ($m > 0$). For a strongly monotone Lipschitz relation,

$$\forall x, y \ m \|x - y\|^2 \leq (fx - fy)^T(x - y) \stackrel{\text{CS}}{\leq} L \|x - y\|^2,$$

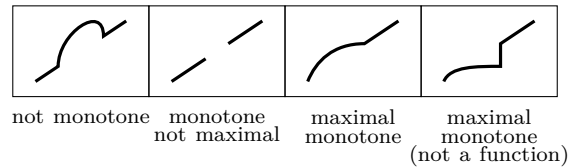
the condition number is $\kappa = L/m$.

If F is maximal monotone, then so is F^{-1} .

If F is strongly monotone with parameter m , the F^{-1} is Lipschitz with constant $1/m$ (but if F is Lipschitz, F^{-1} need not be strongly monotone).

If f is CCP, then ∂f is maximal monotone.

A CCP f is strongly convex if $f(x) - m\|x\|^2$ is convex or, equivalently, if ∂f is strongly monotone with parameter m . A CPP f is strongly convex with parameter m iff f^* is strongly smooth with parameter $L = 1/m$.

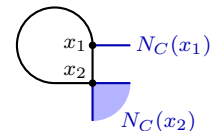


Continuous monotone functions are maximal.

An affine function $f(x) = Ax + b$ is maximal monotone iff $A + A^T \succeq 0$; it is strongly monotone with parameter $\lambda_{\min}(A + A^T)/2$; it is the subdifferential of a CCP iff $A = A^T$ and $A \succeq 0$. This generalizes to differentiable functions $F : \mathbf{R}^n \rightarrow \mathbf{R}^n$ and their derivative $A = DF(x)$.

Projections are monotone. If $C \subset \mathbf{R}^n$ is closed convex, its *normal cone operator* $N_C = \partial I_C$ is

$$N_C(x) = \begin{cases} \{y : \forall z \in C \ y^T(z - x) \leq 0\} & \text{if } x \in C \\ \emptyset & \text{otherwise} \end{cases}$$



The *saddle subdifferential* of $f : \mathbf{R}^m \times \mathbf{R}^n \rightarrow \mathbf{R} \cup \{\infty\}$ is

$$F(x, y) = \begin{pmatrix} \partial_x f(x, y) \\ -\partial_y f(x, y) \end{pmatrix};$$

if f is convex in x and concave in y , F is often maximal.

For the optimization problem

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & f_0(x) \\ \text{Such that} & \forall i \ f_i(x) \leq 0 \quad \text{CCP} \\ & \forall j \ h_j(x) = 0 \quad \text{affine} \end{array}$$

the KKT operator is

$$T(x, \lambda, \nu) = \begin{pmatrix} \partial_x L(x, \lambda) \\ -F(x) + N_{\lambda \geq 0} \\ -H(x) \end{pmatrix}$$

where $F = (f_1, \dots, f_m)$, $H = (h_1, \dots, h_p)$,

$$L(x, \lambda, \nu) = \begin{cases} f_0(x) + \sum \lambda_i f_i(x) + \sum \nu_j h_j(x) & \text{if } \lambda \geq 0 \\ -\infty & \text{otherwise} \end{cases}$$

it is monotone, and its zero set is the set of optimal primal-dual pairs.

5a. Fixed point iterations of a *contraction* converge, at least geometrically. If f is strongly convex and strongly smppth with parameters m and L , *gradient descent*

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \quad \alpha \in (0, 2L)$$

converges to a minimizer of f . If F is strongly monotone and Lipschitz with parameters m and L , $(I - \alpha F)$ is a contraction for $\alpha \in (0, 2m/L^2)$, and the *forward step method* converges

$$x_{k+1} = x_k - \alpha F(x_k).$$

5b. Iterations of *averaged operators* converge to a solution, if one exists (for non-expansive, but non-averaged operators, they need not converge). If f is convex (not strongly) and Lipschitz, $I - \alpha \nabla f$ is averaged, and gradient descent converges.

5c. The *dual ascent* for

$$\text{Minimize}_x f(x) \text{ such that } Ax = b$$

is gradient ascent on the dual, $\text{Minimize}_y g(y)$ where $g(y) = -(f^*(-A'y) - y'b)$,

$$\begin{aligned} x_{k+1} &= \text{Argmin}_x L(x, y_k) \\ y_{k+1} &= y_k + \alpha(Ax_{k+1} - b) \end{aligned}$$

The *convex feasibility problem* looks for $x \in C \cap D$, where C and D are non-empty, closed, convex.

$$\text{Minimize}_x \theta d^2(x, C) + (1 - \theta) d^2(x, D)$$

The gradient method gives

$$\begin{aligned} x_{k+1}^C &= \Pi_C(x_k) \\ x_{k+1}^D &= \Pi_D(x_k) \\ x_{k+1} &= \theta x_{k+1}^C + (1 - \theta) x_{k+1}^D \end{aligned}$$

6a. The **resolvent** and **Cayley operator** of a relation on \mathbf{R}^n are

$$\begin{aligned} R &= (I + \alpha A)^{-1} \\ C &= 2R - I. \end{aligned}$$

If A is monotone, R and C are non-expansive. If A is maximal monotone, $\text{dom } R = \text{dom } C = \mathbf{R}^n$ (fixed point iterations are always defined); $0 \in A(x)$ iff $x = R_A(x)$ iff $x = C_A(x)$. If A is strongly monotone, then R is Lipschitz with $L = 1/(1 + \alpha m)$. If A is strongly monotone and Lipschitz, then C is a contraction with

$$L = \left(1 - \frac{4\alpha m}{(1 + \alpha L)^2}\right)^{1/2}.$$

If A is maximal and single-valued, $C = (I - \alpha A)(I + \alpha A)^{-1}$.

If A is a symmetric matrix with positive eigenvalues, R has eigenvalues in $(0, 1]$, C in $(-1, 1]$.

The resolvent of the subdifferential of a convex function f is the *proximal operator*

$$R(x) = (I + \alpha \partial f)^{-1}(x) = \underset{u}{\text{Argmin}} f(u) + \frac{1}{2\alpha} \|u - x\|^2.$$

For the normal cone operator $N_C(x) = \partial I_C(x)$,

$$\begin{aligned} R &= (I + \alpha N_C)^{-1} = \Pi_C && \text{projection} \\ C &= 2\Pi_C - I - Q_C && \text{overprojection.} \end{aligned}$$

To solve $0 \in A(x)$, where A is maximal monotone, the *Cayley* and *proximal point* methods iterate

$$\begin{aligned} x_{k+1} &= C(x_k) && (\text{may not converge}) \\ x_{k+1} &= R(x_k) && (\text{converges: } R \text{ is averaged}). \end{aligned}$$

6b. To solve $\text{Minimize}_x f(x)$ s.t. $Ax = b$, where f is CCP, the *method of multipliers* looks for a dual variable y such that $0 \in -\nabla g$ with the *proximal point method*, $y \leftarrow R_{-\nabla g}(y)$.

$$\begin{aligned} x_{k+1} &= \text{Argmin}_x L_\alpha(x, y_k) \\ y_{k+1} &= y_k + \alpha(Ax_{k+1} - b). \end{aligned}$$

The *proximal method of multipliers* uses the KKT operator T instead, $(x, y) \leftarrow R_T(x, y)$:

$$\begin{aligned} x_{k+1} &= \text{Argmin}_x L_{\alpha_1}(x, y_k) + \frac{\alpha_2}{2} \|x - x_k\|^2 \\ y_{k+1} &= y_k + \alpha_1(Ax_{k+1} - b). \end{aligned}$$

To find a zero of $F(x) = Ax - b$, where $A + A^T \succ 0$, so that F is maximal monotone, the proximal point method with $R_F = (I + \varepsilon^{-1}F)^{-1}$ gives the *iterative refinement*

$$\begin{aligned} r_k &= Ax_k - b \\ x_{k+1} &= x_k - (A + \varepsilon I)^{-1} r_k \end{aligned}$$

$(A + \varepsilon I)$ is often better conditioned than A).

If F is maximal monotone, then so is $L^{-1}TFL^{-1}$; the corresponding proximal point method (*generalized proximal point method*) is $x_{k+1} = (A + \alpha F)^{-1}Ax_k$, where $A = L^T L \succ 0$.

7a. To solve $0 \in (A + B)(x)$, where A and B are maximal monotone, *forward-backward operator splitting* uses the fixed point iteration $x = (I + \alpha B)^{-1}(I - \alpha A)(x)$, i.e. $x_{k+1} = R_B(x_k - \alpha Ax_k)$; for $\alpha \in (0, 2/L)$ (if A is L -Lipshitz) or $\alpha \in (0, 2m/L^2)$; it is averaged.

The *proximal gradient method* solves $\text{Minimize}_x f(x) + g(x)$, f, g CCP, by applying forward-backward splitting to $0 \in (\partial f + \partial g)(x)$, assuming f is differentiable,

$$x_{k+1} = \underset{x}{\text{Argmin}} f(x_k) + \nabla f(x_k)^T(x - x_k) + g(x) + \frac{1}{2\alpha} \|x - x_k\|^2$$

Forward-backward-forward splitting writes $0 \in (A + B)(x)$ as

$$x = ((I - \alpha A)R_B(I - \alpha A) + \alpha A)(x), \quad \alpha \in (0, 1/L),$$

which gives

$$\begin{aligned} x_{k+\frac{1}{2}} &= R_B(x_k - \alpha Ax_k) \\ x_{k+1} &= x_{k+\frac{1}{2}} - \alpha(Ax_{k+\frac{1}{2}} - Ax_k) \end{aligned}$$

For $B = 0$, this is the *extra gradient* method

$$\begin{aligned} x_{k+\frac{1}{2}} &= x_k - \alpha Ax_k \\ x_{k+1} &= x_k - \alpha Ax_{k+\frac{1}{2}}. \end{aligned}$$

The *Combettes-Pesquet* method solves $0 \in (A + B + C)(x)$ with forward-backward-forward splitting by writing it as

$$0 \in \begin{pmatrix} Ax \\ B^{-1}u \end{pmatrix} + \begin{pmatrix} u + Cx \\ -x \end{pmatrix}.$$

Peaceman-Rachford splitting solves $0 \in (A + B)(x)$ using $C_A C_B(z) = z$, $x = R_B(z)$.

$$\begin{aligned} x_{k+\frac{1}{2}} &= R_B z_k \\ z_{k+\frac{1}{2}} &= 2x_{k+\frac{1}{2}} - z_k \\ x_{k+1} &= R_A(z_{k+\frac{1}{2}}) \\ z_{k+1} &= 2z_{k+\frac{1}{2}} - z_{k+\frac{1}{2}} \end{aligned}$$

Douglas-Rachford splitting uses $(\frac{1}{2}I + \frac{1}{2}C_A C_B)(z) = z$, $x = R_B(z)$, i.e.,

$$\begin{aligned} x_{k+\frac{1}{2}} &= R_B(x_k) \\ z_{k+\frac{1}{2}} &= 2x_{k+\frac{1}{2}} - z_k \\ x_{k+1} &= R_A(z_{k+\frac{1}{2}}) \\ z_{k+1} &= z_k + x_{k+1} - x_{k+\frac{1}{2}}. \end{aligned}$$

For instance, to solve $\text{Minimize}_x \sum f_i(x)$, rewrite it as

$$\text{Minimize}_{x_1, \dots, x_n} \sum f_i(x_i) \text{ s.t. } x_1 = x_2 = \dots = x_n,$$

then, as

$$0 \in \begin{pmatrix} \partial f_1(x_1) \\ \vdots \\ \partial f_n(x_n) \end{pmatrix} + N_{[x_1 = \dots = x_n]}(x_1, \dots, x_n),$$

which gives

$$\begin{aligned} x_i &\leftarrow \underset{x}{\text{Argmin}} f_i(x) + \frac{1}{2\alpha} \|x - z_i\|^2 \\ z_1 &\leftarrow z_i + 2\bar{x} - \bar{z} - x_i. \end{aligned}$$

Davis-Yin 3-operator splitting solves $0 \in (A + B + C)(x)$, where A, B, C are maximal monotone, by writing it as $Tz = z$, $x = R_B(z)$, where $T = C_A(C_B - \alpha C)R_B - \alpha CR_B$; the iteration is

$$\begin{aligned} x_{k+\frac{1}{2}} &= R_B(z_k) \\ z_{k+\frac{1}{2}} &= 2x_{k+\frac{1}{2}} - z_k \\ x_{k+1} &= R_A(z_{k+\frac{1}{2}} - \alpha Cx_{k+\frac{1}{2}}) \\ z_{k+1} &= z_k + x_{k+1} - x_{k+\frac{1}{2}} \end{aligned}$$

7b. The proximal gradient solves $\text{Minimize}_x f(x) + \lambda \|x\|_1$, where f is CCP, as (*iterative shrinkage thresholding algorithm*, ISTA)

$$\begin{aligned} x &\leftarrow S_{\alpha\lambda}(x - \alpha \nabla f(x)) \\ S_k(x)_i &= \text{sign}(x_i)(|x_i| - \kappa)_+. \end{aligned}$$

The proximal gradient solves $\text{Minimize}_{x \in C} f(x)$, f CCP, C convex, as (*projected gradient*)

$$x \leftarrow \Pi_C(x - \alpha \nabla f(x)).$$

The proximal gradient solves the convex feasibility problem $x \in C \cap D$, i.e., $\text{Minimize}_{x \in C} d(x, D)^2$, as (*alternating projections*)

$$x \leftarrow \Pi_C \Pi_D x.$$

Douglas-Rachford splitting solves it as $0 \in (N_C + N_D)(x)$

$$\begin{aligned} x_{k+\frac{1}{2}} &= \Pi_D(z_k) \\ x_{k+1} &= \Pi_C(2x_{k+\frac{1}{2}} - z_k) \\ z_{k+1} &= z_k + x_{k+1} - x_{k+\frac{1}{2}}. \end{aligned}$$

Chambolle-Pock solves $\text{Minimize}_x f(x) + g(Mx)$ by rewriting it as

$$0 \in \begin{pmatrix} \partial f(x) \\ \partial g^*(u) \end{pmatrix} + \begin{pmatrix} 0 & M^T \\ -M & 0 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix}$$

and applying the generalized proximal point method with $A = \begin{pmatrix} I & -\alpha M^T \\ -\alpha M & I \end{pmatrix}$

$$\begin{aligned} x_{k+1} &= R_{\partial f}(x_k - \alpha M^T u_k) \\ u_{k+1} &= R_{\partial g^*}(u_k + \alpha M(2x_{k+1} - x_k)) \end{aligned}$$

(if R_{g^*} is easier to compute than $R_P M^T \partial g M$).

The linear program $\text{Minimize}_{x \geq 0} c^T x$ s.t. $Ax = b$ can be written as $0 \in (T_1 + T_2)(x, \nu, \lambda)$, where

$$T_1(x, \nu, \lambda) = \begin{pmatrix} x + A^T \nu - \lambda \\ -(Ax - b) \\ x \end{pmatrix} \quad T_2(x, \nu, \lambda) = \begin{pmatrix} 0 \\ 0 \\ N_{[\lambda \geq 0]} \end{pmatrix}$$

and solved with forward-backward-forward splitting.

The convex-concave game $\text{Maximize}_y \text{Minimize}_x f(x, y)$ can be written $0 \in F(x, y)$, where

$$F(x, y) = \begin{pmatrix} \partial_x f(x, y) \\ \partial_y - f(x, y) \end{pmatrix}$$

and solved with forward-backward-forward splitting (*extra gradient*)

$$\begin{aligned} x_{k+\frac{1}{2}} &= x_k - \alpha \nabla_x f(x_k, y_k) \\ y_{k+\frac{1}{2}} &= y_k + \alpha \nabla_y f(x_k, y_k) \\ x_{k+1} &= x_k - \alpha \nabla_x f(x_{k+\frac{1}{2}}, y_{k+\frac{1}{2}}) \\ y_{k+1} &= y_k + \alpha \nabla_y f(x_{k+\frac{1}{2}}, y_{k+\frac{1}{2}}) \end{aligned}$$

The *complementarity problem*

$$\begin{aligned} \text{Find} \quad & x \in K \\ \text{Such that} \quad & F(x) \in K^* \\ & x \perp F(x) \end{aligned}$$

(with F maximal monotone and Lipschitz) can be written as $0 \in (F + N_K)(x)$ and solved with forward-backward-forward splitting

$$\begin{aligned} x_{k+\frac{1}{2}} &= \Pi_K(x_k - \alpha F x_k) \\ x_k &= x_{k+\frac{1}{2}} - \alpha(F x_{k+\frac{1}{2}} - F x_k). \end{aligned}$$

Quasidefinite systems

$$Kx = b, \quad K = \begin{pmatrix} -A & C \\ C^T & B \end{pmatrix}, \quad A, B \succ 0$$

can be written as

$$F(x) = 0, \quad F(x) = J(Kx - b), \quad J = \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix},$$

split using

$$K_1 = \begin{pmatrix} -A & 0 \\ 0 & B \end{pmatrix}, \quad K_2 = \begin{pmatrix} 0 & C \\ C^T & 0 \end{pmatrix},$$

and solved with Peaceman-Rachford.

ADMM solves

$$\begin{aligned} \text{Find} \quad & x, z \\ \text{To minimize} \quad & f(x) + g(z) \\ \text{Such that} \quad & Ax + Bz = c \end{aligned}$$

by applying Douglas-Rachford to the dual problem

$$\text{Maximize}_{\nu} -f^*(-A^T \nu) - g^*(-B^T \nu) + c^T \nu$$

(there are other derivations of ADMM).

Parameter selection and pre-conditioning for a graph form solver C. Fougner and S. Boyd (2015)

The *graph form* optimization problem

$$\begin{aligned} \text{Find} \quad & x, y \\ \text{To minimize} \quad & f(y) + g(x) \\ \text{Such that} \quad & y = Ax \end{aligned}$$

can be solved with ADMM

$$\begin{aligned} (x', y') &= \text{prox}_g(x - \tilde{x}), \text{prox}_f(y - \tilde{y}) \\ (x, y) &= \Pi(x' + \tilde{x}, y' + \tilde{y}) \\ (\tilde{x}, \tilde{y}) &= (\tilde{x} - (x - x'), \tilde{y} - (y - y')). \end{aligned}$$

This can be sped up with

- Over-relaxation: replace x' with $\alpha x' + (1 - \alpha)x$, $\alpha \in (1.5, 1.8)$;
- Approximate projection
- Preconditioning: replace y , x and A with Dy , $E^{-1}x$ and DAE respectively, such that the singular values of DAE be close to 1

$$\text{cond}(DAE) \approx 1, \quad \|DAE\|_2 \approx 1;$$

- Adaptive proximal penalty ρ ,

$$\text{prox}_f v = \underset{z}{\text{Argmin}} f(z) + \frac{\rho}{2} \|x - v\|_2^2$$

(start with a large ρ , which encourages primal feasibility, then, when the primal variables (x, y) converge, decrease it; when the dual variables (\tilde{x}, \tilde{y}) converge, increase it again).

Applications include portfolio optimization: $w' \mu - \lambda w'(FF' + D)w$ becomes

$$\begin{aligned} g(x) &= x' \mu - \lambda x' D x + I(x \geq 0) \\ f(y) &= \lambda y' y + I(y_{m+1} = 1) \\ y &= \begin{pmatrix} F' \\ \mathbf{1}' \end{pmatrix} x. \end{aligned}$$

C++ implementation: `pogs`.

Pretrained transformers for text ranking: BERT and beyond J. Lin at al. (2021)

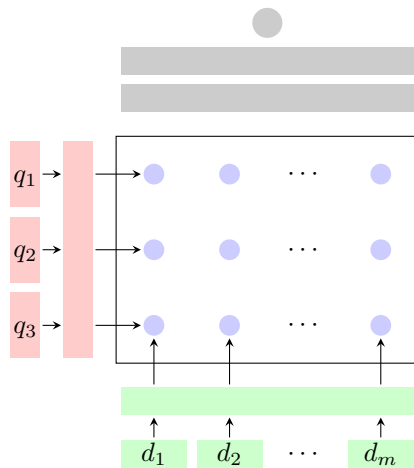
Information retrieval is often done in two steps: first identify a set of documents “close” to the query, with a bag-of-words approach such as BM25 (lucene)

$$\text{BM}_{25}(q, d) = \sum_{t \in q \cap d} \log \frac{N - \text{df}_t + .5}{\text{df}_t + .5} \cdot \frac{\text{df}_{d,t}(k+1)}{\text{tf}_{d,t} + k \cdot (1 - n + b \ell_d / L)}$$

(where ℓ_d is the length of document d and L the average document length), second, rerank those documents, e.g., using

- Cosine similarity between query and document embeddings

- An **interaction model**, *i.e.*, a neural network exploiting the histogram of similarities between query terms and document terms



(You can stack several rerankers.)

3. BERT is a transformer-based encoder-only network (GPT is decoder-only; the original transformer is a sequence-to-sequence model) providing *contextual embeddings*.

MonoBERT uses BERT to predict if a document is relevant to a query

$$P[\text{relevant} \mid [\text{CLS}] q [\text{SEP}] d [\text{SEP}]]$$

from the contextual embedding of the [CLS] token.

To overcome the input length limit:

- Birch trains BERT on short passages (e.g., tweets), and uses aggregated sentence scores;
- VERT-MaxP trans BERT on overlapping passages, assuming that they are all relevant (or irrelevant) if the document is, and uses aggregated (max) passage relevance at inference time;
- CEDR feeds BERT embeddings to an interaction model;
- Parade uses BERT to compute a vector representation of each sentence, then another BERT to turn that sequence of sentence embeddings into a document embedding.

The Reformer (LSH attention) and the Longformer (sparse attention) also deal with long documents,

DuoBERT compares the relevance of two documents.

4. The vocabulary used in a query may not match that in the document (e.g., “car”, vs “automobile”):

- The query can be expanded:
 - First, retrieve documents with the query words;
 - Assume the first results are relevant;
 - Add words from those documents to the query;
 - Retrieve more documents;

Those words can be chose by training a model

$$P[\text{term} \mid \text{query}, \text{document}]$$

measuring how relevant a term is (CEQE);

- The query can be reformulated to make it more human-like (BERT works better with those);
- The document themselves can be turned into queries (doc2query, trained on MS MARCO);
- The query terms can be re-weighted, by modeling

$$\text{representation of document } d \text{ term } t \mapsto \frac{\# \text{ queries relevant for } d \text{ with } t}{\# \text{ queries relevant for } d}$$

- Expansion and reweighting can be combined.

5. Document retrieval, with approximate neighbour (ANN) search, from pretrained dense representations, is worse than retrieval from average pooling of GloVe embeddings – fine-tuning is necessary, and careful selection of negative samples improves performance.

InPars: data augmentation for information retrieval using large language models
L. Bonifacio et al.

Given a (small) set of (query, document) pairs, generate plausible queries for all the documents (few shot learning, with GPT3 and the MS MARCO dataset, providing triplets text / good question / bad question); keep the highest probability pairs, and use them for reranking (T5, after pyserini).

Pyserini: a Python toolkit for reproducible information retrieval research with sparse and dense representations
J. Lin et al.

ToxiGen: a large-scale machine-generated dataset for adversarial and implicit hate speech detection
T. Hartvigsen et al.

Toxic language detection systems often incorrectly flag texts merely because they mention some minority group (often targeted by online hate): use a language to generate subtly toxic and benign text about those minorities.

Zero-shot neural passage retrieval via domain-targeted synthetic question generation
J. Ma et al.

Synthetic question generation, to compute question embeddings (instead of document embeddings), for first stage information retrieval (nearest neighbour).

SwinIR: image restoration using Swin transformer
J. Liang et al.

Transformers for image restoration (upsampling, denoising, artefact removal):

- Start with a 3×3 convolution, to extract shallow features;
- Apply 6 residual swin transformer blocks (RSTB), followed by a convolution;

- Each RSTB has 6 swin transformer layers (STL), followed by a convolution;
- Each STL has a LayerNorm, a multihead self-attention (MSA) on non-overlapping $M \times M$ windows, another LayerNorm, an MLP;
- The network ends with a convolution, to reconstruct the image from the shallow and deep features (there are skip-connections everywhere).

Generalized cross-entropy loss for training deep neural networks with noisy labels
Z. Zhang and M.R. Sabuncu

To increase robustness to noisy labels, try the negative Box-Cox transformation $\text{loss}(j, p) = (1 - p_j^q)/q$, $q \in (0, 1)$, where j is the label (y) and p the probabilities output by the model (\hat{y}). The cross-entropy $\ell = \log p_j$ and the MAE $\ell = \|e_j - p\|_1 = 2 - 2p_j$ are limiting cases, for $q \rightarrow 0$ or 1. Truncate by replacing p_j with $\text{Min}(p_j, k)$.

Modeling tabular data using conditional GAN
L. Xu et al. (2019)

To account for multimodal continuous columns, estimate the number of modes with a *variational Gaussian mixture* (VGM) and fit a Gaussian mixture; then, separately sample the mixture component and the value within that component. To account for the imbalance in discrete columns,

- Learn conditional generators, conditioned on the value of one of the discrete columns,
- Sample the discrete values according to the log-frequency of each category.

Opacus: user-friendly differential privacy library in pytorch
A. Yousefpour et al. (2021)

DP-SGD (differentially private stochastic gradient descent) makes SGD differentially private (intuitively, removing one sample from the batch does not change the update) by computing the per-sample gradients, clipping them, aggregating them, and adding noise.

Supervised contrastive learning
P. Khosla et al. (2020)

The *triplet loss* uses one positive sample (augmentation) and one negative sample, but requires negative mining. *Contrastive learning* uses one positive sample and many negative samples

$$\text{loss} = \sum_i -\log \frac{\exp(z_i \cdot z_{j(i)}/\tau)}{\sum_{a \in N(i)} \exp(z_i \cdot z_a/\tau)}$$

i : samples

$j(i)$: positive sample

$N(i)$: negative samples

$z_i \in \mathbf{S}^{d-1}$: features.

Supervised contrastive learning uses many positives (augmentations and samples from the same class) and many negative samples

$$\text{loss} = \sum_i -\text{Mean}_{p \in P(i)} \log \frac{\exp(z_i \cdot z_p/\tau)}{\sum_{a \in N(i)} \exp(z_i \cdot z_a/\tau)}$$

(putting the mean outside the log rather than inside works better).

Generate features in 2048 dimensions, normalize them, reduce the dimension to 128 (MLP), and normalize again; use scalar products instead of Euclidean distance.

Energy-latency attacks via sponge poisoning
A.E. Cinà (2022)

Mount a DoS attack on a deep learning model, running on sparsity-based ASIC accelerators adopting zero-skipping strategies to avoid multiplications when the input is zero, by adding a penalty (ℓ_0 relaxation)

$$\hat{\ell}_0(x) = \sum \frac{x_i^2}{x_i^2 + \varepsilon}$$

for non-zero activations.

A new correlation coefficient between categorical, ordinal and interval variables with Pearson characteristics
M. Baak et al. (2019)

The ϕ_k correlation coefficient (implementation in **phik**) is the correlation for which a Gaussian variable would have the same χ^2 (after binning the quantitative variables into 10 bins).

The *global correlation coefficient* is the highest possible correlation between variable k and a linear combination of the other variables.

$$g_k = \sqrt{1 - [V_{kk} \cdot (V^{-1})_{kk}]^{-1}}$$

Decision tree learning with spatial modal logics
G. Pagliarini and G. Sciacicco

Modal logic extends propositional logic with

$$\begin{array}{ll} \Diamond P & \text{possibly } P \\ \Box P & \text{necessarily } P. \end{array}$$

It is less expressive than first order logic (FOL, which allows quantifiers, \forall, \exists).

Allen's interval algebra



can be generalized to hyperrectangles, or replaced with the *region connection calculus* (RCC8)



Modal decision trees, for temporal or spatial data, allows rules like

$$\begin{aligned} \exists[a, b] \ T_{[a, b]} \geq 39 & \quad \exists R_1 \ \text{Red}(R_1) \geq a \\ \exists[c, d] \ \text{HR}_{[c, d]} \geq 140 & \quad \text{or} \quad \exists R_2 \ \text{Green}(R_2) \leq b \\ [a, b] \cap [c, d] \neq \emptyset & \quad R_2 \subset R_1. \end{aligned}$$

They are more expressive (and sparser) than decision trees, but more interpretable than ILP models (inductive logic programming, *i.e.*, FOL rules).

**Bayesian streaming
sparse Tucker decomposition**
S. Fang et al. (2021)

The CP decomposition is a Tucker decomposition with a diagonal core – instead, add a sparsifying penalty (with a Bayesian spike-and-slab prior).

**CoSTCo: a neural tensor completion model
for sparse tensors**
H. Liu et al. (2019)

The Tucker decomposition

$$T_{i_1, \dots, i_N} = \sum_{j_1, \dots, j_N} G_{j_1, \dots, j_N} U_{j_1 i_1}^1 U_{j_2 i_2}^2 \dots U_{j_N i_N}^N$$

can be made nonlinear $T_{i_1, \dots, i_N} = f_\theta(i_1, \dots, i_N)$:

- Use the 1-hot encoding $e \in \mathbf{R}^{\sum d_i}$ of the index tuple (i_1, \dots, i_N) ;
- Compute an embedding

$$\begin{pmatrix} U^1 & & 0 \\ & \ddots & \\ 0 & & U^N \end{pmatrix} e \in \mathbf{R}^{r \times N};$$

- Apply CNNs with filter sizes $(r, 1)$ and $(1, N)$;
- Finish with a fully-connected layer.

**How many bins should be put
in a regular histogram**
L. Birgé and Y. Rozenholc (2005)

Choose the number of bins D to partition $[0, 1]$ by maximizing

$$\begin{aligned} L(D) &= \sum_{1 \leq j \leq D} N_j \log \frac{DN_j}{n} - \text{pen}(D) \\ \text{pen}(D) &= D - 1 + (\log D)^{2.5}. \end{aligned}$$

Test with $N(0, 1)$, $\frac{1}{4}N(0, 1) + \frac{3}{4}N(2, \frac{1}{16})$, $\frac{3}{4}N(0, 1) + \frac{1}{4}N(3, \frac{1}{9})$, $\text{Exp}(1)$, $T(3)$, $U(0, 1)$.

**Cone-constrained monotone mean-variance
portfolio selection under diffusion models**
Y. Shen and B. Zou (2022)

The mean-variance preference

$$J(X) = \mathbb{E}_{\mathbf{P}}[X] - \lambda \text{Var}_{\mathbf{P}}[X]$$

is not monotonic: we can have $X \geq X'$ a.s., but $J(X) < J(X')$ (for instance $X = \bullet\bullet\bullet\bullet\bullet$, $X' = \bullet\bullet\bullet\bullet\bullet$, $\lambda = 1$)

Replace it with the monotone mean variance preference

$$J(X) = \inf_{\mathbf{Q} \ll \mathbf{P}} \mathbb{E}_{\mathbf{Q}} \left[X + \lambda^{-1} \left(\frac{d\mathbf{Q}}{d\mathbf{P}} - 1 \right) \right].$$

**Portfolio selection with monotone
mean-variance preferences**
F. Maccheroni et al. (2009)

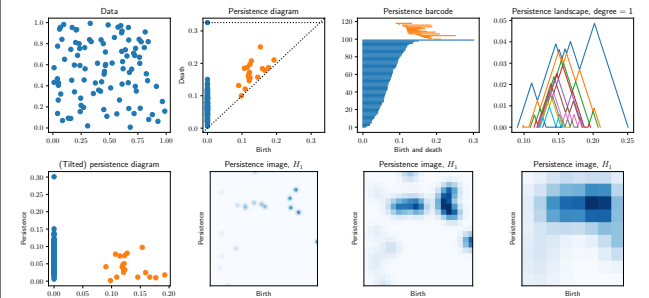
The MMV preference coincides with the mean-variance preference on its domain of monotonicity (the subset of L^2 where the Gâteaux derivative of $U(f) = \mathbb{E}[f] - \frac{1}{2}\lambda \text{Var}[f]$ is positive), and is the best monotone approximation outside.

The unconstrained optimal portfolio is still of the form $\alpha^* = \lambda^{-1}V^{-1}\mu$, but V and μ are now *lower* moments of X (for some threshold).

**Persistence images: a stable vector
representation of persistent homology**
H. Adams et al. (2017)

Persistence diagrams can be fed directly to distance-based machine learning algorithms (with the bottleneck distance, or the Wasserstein distance), but not to arbitrary machine learning algorithms (which often require coordinates). The *persistence image* is a vector representation for a persistence diagram (PD) obtained by:

- Rotating the (birth, death) PD into (birth, persistence), where persistence = death – birth;
- Replacing each point with a Gaussian distribution (you need to choose its variance);
- Considering a weighted sum of those distributions, with weights increasing with persistence (it should be zero on the diagonal) \lhd
- Rasterising the data (for H_0 , the birth coordinate is always zero: it is a 1-dimensional image).



**Persistence diagrams
with linear machine learning models**
I. Obayashi and Y. Hiraoka

Linear models built on the *persistence image* are interpretable: large positive or negative coefficients correspond to desired or undesirable regions of the persistence diagram.

Add a regularization term to smooth the weights for higher resolution persistence images.

***Persistence weighted Gaussian kernel
for topological data analysis***
G. Kusano et al.

Map persistence diagrams D to functions

$$D \mapsto \sum_{x \in D} w(x)k(x, \cdot)$$

where k is a kernel and w a weight function decreasing with persistence.

***Persistent entropy for separating topological
features from noise in Vietoris-Rips complexes***
N. Atienza et al. (2017)

Given a persistence diagram $(b_i, d_i)_i$, the *persistent entropy* is the entropy H of the persistences $p_i \propto d_i - b_i$; it can help prune non-significant barcodes (intuitively, $\exp H$ is the number of significant barcodes).

***Tropical coordinates
on the space of persistence barcodes***
S. Kališnik

The *tropical semiring* is $(\bar{\mathbf{R}}, \text{Min}, +, \infty, 0)$; the *arctic semiring* is $(\mathbf{R} \cup \{-\infty\}, \text{Max}, +, -\infty, 0)$. Max-plus polynomials up to functional equivalence (identify polynomials if they always take the same values) form a semiring $\text{MaxPlus}[x_1, \dots, x_n]$. Similarly, Min-plus polynomials, and rationals tropical functions, form semirings.

Symmetric tropical rational functions can be used as coordinates on the barcode space.

***The ring of algebraic functions
on persistence barcodes***
A. Adcock et al. (2013)

We could use symmetric polynomials as coordinates on the space of barcodes: $k[x_1, y_1, \dots, x_n, y_n]^{\mathfrak{S}_n}$ is complicated (there are non-trivial relations between the symmetric polynomials – syzygies), but $\varinjlim k[x_1, y_1, \dots, x_n, y_n]^{\mathfrak{S}_n}$ is freely generated. However, we do not only want $(x_1, y_1, \dots, x_n, y_n) \sim (x_1, y_1, \dots, x_n, y_n, 0, 0)$, but $(x_1, y_1, \dots, x_n, y_n) \sim (x_1, y_1, \dots, x_n, y_n, x_{n+1}, x_{n+1})$: the corresponding subring of $k[x_1, y_1, \dots, x_n, y_n]^{\mathfrak{S}_n}$ is that of polynomials f such that

$$\left(\frac{\partial}{\partial x_i} + \frac{\partial}{\partial y_i} \right) f \in (y_i - x_i)$$

and the limit of those subrings is freely generated by the

$$\sum_i x_i^{a+1} y_i^b - x_i^a y_i^{b+1}.$$

(Those coordinates are not Lipschitz wrt the bottleneck or Wasserstein distance.)

***Stable topological signatures
for points on 3D shapes***
M. Carrière et al.

To identify a point on a surface, consider their persistence diagram, *i.e.*, the 1-holes of the geodesic balls centered on this point.

To get a vector embedding, compute the pairwise distances between the points in the persistence diagram, sort them, and pad them with zeroes,

$$m(x, y) = \text{Min}\{d(x, y), d(x, \text{diag}), d(y, \text{diag})\}.$$

***A stable multi-scale kernel
for topological machine learning***
J. Reininghaus et al.

Define a feature map $\Phi_\sigma : \text{PD} \rightarrow L^2$ on the space of persistence diagrams by replacing each point p with a Gaussian density and subtracting another Gaussian \bar{p} on the other side of the diagonal.

$$\begin{aligned} \Phi_\sigma(D) : x &\mapsto \sum_{p \in D} \phi_\sigma(\|x - p\|) - \phi_\sigma(\|x - \bar{p}\|) \\ \phi_\sigma(t) &= e^{-t^2/\sigma^2} \end{aligned}$$

The corresponding kernel is

$$k_\sigma(F, G) = \sum_{\substack{p \in F \\ q \in G}} \phi_\sigma(\|p - q\|) - \phi_\sigma(\|p - \bar{q}\|).$$

***Statistical analysis
of persistence intensity functions***
Y.C. Chen et al.

Replace persistence diagrams with weighted sums of Gaussians, one for each point, with weights proportional to the persistence, $w = \text{death} - \text{birth}$: you can then easily average them, cluster them, etc.

A topology layer for machine learning
R. Brüel-Gabrielsson et al. (2020)

Persistence diagrams as a layer: you can use

$$\xi(p, q, i_0, \text{PD}_k) = \sum_{i \geq i_0} |d_i - b_i|^p \left(\frac{b_i + d_i}{2} \right)^q$$

in the loss, *i.e.*, increase or decrease $\beta_0, \beta_1, (\beta_0 - 1)_+, (\beta_1 - 1)_+$, etc.

Use as a regularizer (image reconstruction) or a prior (generative models).

The Bayesian learning rule
M.E. Khan and H. Rue

Empirical risk minimization

$$\text{Minimize}_{\theta} \sum_i \ell(y_i, f_{\theta}(x_i)) + R(\theta)$$

can be generalized to probability distributions

$$\text{Minimize}_q \mathbb{E}_{\theta \sim q} \left[\sum_i \ell(y_i, f_{\theta}(x_i)) \right] + \text{KL}(q(\theta) \parallel p(\theta))$$

$$p(\theta) \propto \exp -R(\theta).$$

The *Bayesian learning rule* looks for a probability distribution in an exponential family

$$q(\theta) = h(\theta) \exp[\langle \lambda, T(\theta) \rangle - A(\lambda)]$$

λ : natural parameters
 A : log-partition (cummulant) function
 $T(\theta)$: sufficient statistics
 h : base measure
 $\mu = \mathbb{E}_{\theta \sim q} [T(\theta)]$ expectation parameters

by iterating

$$\lambda_{t+1} \leftarrow \lambda_t - \rho_t \tilde{\nabla}_t \left[\mathbb{E}_{\theta \sim q_t} \bar{\ell}(\theta) - H(q_t) \right]$$

where

$$\bar{\ell}(\theta) = \sum_i \ell(y_i, f_{\theta}(x_i))$$

$$H(q_t) = \mathbb{E}_{\theta \sim q_t} [-\log q_t(\theta)] \text{ entropy}$$

$$\tilde{\nabla}_t \mathbb{E}_{\theta \sim q_t} [\cdot] = F(\lambda_t)^{-1} \nabla_t \mathbb{E}_{\theta \sim q_t} [\cdot] \text{ natural gradient}$$

$$= \nabla_{\mu} \mathbb{E}_{\theta \sim q_t} [\cdot] \Big|_{\mu = \nabla_{\lambda} A(\lambda_t)}$$

$$F(\lambda) = \nabla_{\lambda}^2 A(\lambda) \text{ Fisher information matrix.}$$

For instance, the exponential families $N(m, I)$ (unknown mean, known variance) and $N(m, S^{-1})$ (unknown mean and variance) give gradient descent and Newton's method.

Do t -statistic hurdles need to be raised?
A.Y. Chen (2022)

Calls to lower the 5% significance rate for publications account for false positives but ignores (unobserved) false negatives.

MDD aggregated two-sample test
A. Schrab et al. (2021)

To test if two samples come from the same distribution (non-parametrically, not only in dimension 1, contrary to the Kolmogorov-Smirnov test) use the maximum mean discrepancy (MMD) test (an integral probability metric)

$$\text{MMD}_k(p, q) = \sup_{f \in \mathcal{H}} \left| \mathbb{E}_{X \sim p} f(X) - \mathbb{E}_{Y \sim q} f(Y) \right|$$

$$\|f\|_{\mathcal{H}_k} \leq 1$$

where \mathcal{H}_k is the RKHS associated to a kernel k

$$\widehat{\text{MMD}}_a = \text{Mean}_{i, i'} k(x_i, x_{i'}) + \text{Mean}_{j, j'} k(x_j, x_{j'})$$

$$- 2 \text{Mean}_{ij} k(x_i, x_j)$$

or

$$\widehat{\text{MMD}}_b = \text{Mean}_{i, j} h(X_i, X_j, Y_i Y_j)$$

$$h(x, x', y, y') = k(x, x') + k(y, y') - k(x, y') - k(x', y)$$

JKONet: proximal optimal transport modeling of population dynamics
C. Bunne et al. (2021)

The *JKO flow* is an analogue of proximal descent for probability distributions

$$\rho_{t+1} = \underset{\rho}{\text{Argmin}} J(\rho) + \lambda W^2(\rho_t, \rho)$$

where ρ_t is a probability distribution and W the 2-Wasserstein distance.

The optimal transport map is the gradient of a convex function (if ρ_t has a density – Brenier's theorem), which can be modeled with an *input-convex neural net* (ICNN).

$$T = \nabla \psi_{\theta}$$

$$\rho_{t+1} = (\nabla \psi_{\theta^*})_{\#} \rho_t$$

$$\theta^* = \underset{\theta}{\text{Argmin}} J((\nabla \psi_{\theta})_{\#} \rho_t) + \lambda \int \|x - \nabla \psi_{\theta}(x)\|^2 d\rho_t(x)$$

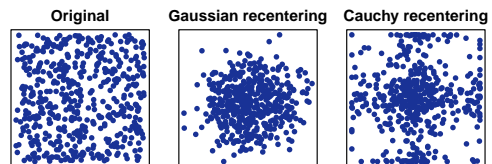
We want to find J such that the JKO flow describes the evolution of a cloud of points

$$J_{\xi}(\rho) = \int_{\xi} \mathbb{E}(x) d\rho(x).$$

Implementation in JAX, with `ott`.

Fully parallel hyperparameter search: reshaped space-filling
M.L. Cauwet et al.

Transform low discrepancy sequences in $[0, 1]^d$ as $x \mapsto \Phi(\lambda \cdot \Phi^{-1}(x))$ or $x \mapsto \Phi(\lambda \cdot \Phi_{\text{Cauchy}}^{-1}(x))$ (remove the outer Φ if the parameters are unbounded).



The *Hammersley sequence* is the same as the Halton sequence, except for the first dimension, in which the points are equidistant (the Halton and Hammersley sequences have a “scrambled” version, with better discrepancy guarantees).

Evolution under strong noise: a self-adaptive evolution strategy can reach the lower performance bound – the pcCMSA-ES

M. Hellwig and H.G. Beyer

Adaptively change the population size in CMSA-ES, increasing it if the optimization stalls because of noise.

Covariance matrix adaptation revisited: the CMSA evolution strategy
H.G. Beyer and B. Sendhoff

CMSA-ES is a simplification of the CMA-ES optimization algorithm with fewer hyperparameters, scaling to larger populations.

Parametric estimate of intensity inhomogeneities applied to MRI
M. Styner et al. (2000)

In “(1+1) evolution strategy”, both the population size and the number of children are equal to 1.

Portfolio performance attribution via Shapley value
N. Moehle et al. (2021)

The Shapley attribution can be defined as

$$a = \frac{1}{n!} \sum_{\pi \in \mathfrak{S}_n} a_\pi$$

where

$$a_{\pi,i} = f(e_{\pi(1)} + \dots + e_{\pi(i)}) - f(e_{\pi(1)} + \dots + e_{\pi(i-1)})$$

and approximated by sampling permutations $\pi \in \mathfrak{S}_n$ (with replacement) but we end up evaluating the same $f(x)$ many times. Instead, use

$$a_i = \sum_{x: x_i=0} \frac{(\mathbf{1}'x)!(n - \mathbf{1}'x - 1)}{n!} (f(x + e_i) - f(x))$$

(where we assume, wlg, $f(0) = 0$) and approximate it

$$a_i = \mathbb{E}_{x \sim p} [f(x + e_i) - f(x)]$$

$$p(x) = \frac{(\mathbf{1}'x)!(n - \mathbf{1}'x - 1)}{n!}$$

by sampling the number of active features $\mathbf{1}'x$ from a multinomial distribution

$$p_k = \frac{k!(n - k - 1)!}{n!}$$

and then sampling $\mathbf{1}'x$ features at random.

Does non-linear factorization of financial returns help build better and stabler portfolios?
B. Spilak and W.K. Härdle (2022)

Statistical factor portfolios can be computed with a low rank factorization

$$X \approx WF$$

X : asset returns

F : factor returns.

Convex non-negative matrix factorization adds constraints

$$W \geq 0, \quad F = H'X, \quad H \geq 0$$

(the assets are non-negative linear combinations of the factors, and conversely).

Non-linear statistical factors can be computed with an autoencoder

$$x \mapsto z = Hx + b \mapsto W\sigma(z) + \varepsilon$$

with constraints $H \geq 0$, $W \geq 0$ and penalties $\|W'W - I\|_2^2$ and $\|\Sigma_z \odot (1 - I)\|^2$.

Use risk parity, hierarchical risk parity, etc. to combine the factor portfolios.

Fast high-dimensional integration using tensor networks
S. Cassel

The tensor train decomposition generalizes the approximation

$$f(x, y) \approx \frac{f(x, \bar{y})f(\bar{x}, y)}{f(\bar{x}, \bar{y})};$$

it interpolates a function from its restrictions on a few 1-dimensional axis-aligned subspaces

$$f(x, y) \approx \sum_{ij} f(x, b_j)(Q^{-1})_{ij} f(a_{ij}, y)$$

$$Q_{k\ell} = f(a_k, b_\ell)$$

In dimension n :

$$f(x_1, \dots, x_q) = F(x_1)Q_{12}^{-1}F_2(x_2) \cdots Q_{d-1,d}^{-1}F_d(x_d).$$

Application include high-dimensional integration; it can be combined with Aitken extrapolation.

Given a sequence $(\psi_n)_n$, Aitken extrapolation writes it as a telescopic series.

$$\psi_n = \psi_0 + \sum_{i=0}^{n-1} (\psi_{i+1} - \psi_i) \quad \text{telescopic series}$$

$$= \psi_0 + \sum_{i=0}^{n-1} (g(\psi_i) - \psi_i)$$

$$g(\psi_\infty) = \psi_\infty \quad \text{fixed point}$$

$$\frac{g(\psi_i)}{\psi_i - \psi_\infty} \approx g'(\psi_i) \quad \text{secant method}$$

$$\psi_\infty \approx \psi_i - \frac{g(\psi_i)}{g'(\psi_i)}$$

Reciprocity in machine learning
M. Sundararajan and W. Krichene (2022)

Compare the average over i and the average over j of the influence of observation i on prediction j (the latter is Cook's distance): they tend to be similar.

Bridging level- k to Nash equilibrium

D. Levin and L. Zhang (2022)

Nash equilibrium assumes that each player is equally sophisticated. A level 0 player plays at random; a level $n + 1$ player plays the best strategy assuming the opponent is level n . An NLK₁ player plays the optimal strategy assuming the opponent is level 0 with probability λ and NLK₁ with probability $1 - \lambda$.

Generative art using neural visual grammars and dual encoders

C. Fernando et al. (2021)

Given a text prompt, generate an image with a hierarchical neural L -system and evaluate them with an image/text dual encoder.

Planting undetectable backdoors in machine learning models

S. Goldwasser et al.

Given a natural training algorithm

$\text{train} : \text{data} \mapsto \text{classifier},$

a *backdoor* is a pair of algorithms

Backdoor : $\text{data} \mapsto \text{classifier, key}$

Activate : $\text{input, key} \mapsto \text{new input}$

such that

- The classifier is indistinguishable from the natural one
- The new input is close to the old, but ends up in the other class.

Undetectable backdoors exist, for dimension reasons (the same arguments proves the existence of adversarial examples) and can easily be built, e.g., for models using random Fourier features.

Manipulating SGD with data ordering attacks

I. Shumailov et al. (2021)

You can poison a model by tampering with the order of the samples used to train it (rank the samples with a surrogate loss).

A generalist agent

S. Reed et al. (2022)

Compress (behaviour cloning) 600 reinforcement learning, language and image models into a single model, with a transformer: convert all the data into discrete sentences (break down images into 16×16 patches, in raster order; mu-law encode

$$F(x) = \text{sign}(x) \frac{\log(1 + \mu |x|)}{\log(1 + \mu M)}$$

continuous values into $[-1, 1]$, and discretize them into 24 bins).

Model cards for model reporting

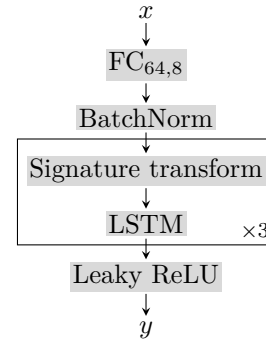
M. Mitchell et al.

The *model card* should include: data, model type, model architecture, training algorithm, loss, intended use, out-of-scope use, performance metrics, performance across different factors or subpopulations.

Deep signature models for financial equity time series prediction

M. Noguer i Alonso et al. (2022)

Forecast time series (daily returns, for 10 US stocks, from 14 days of data), by stacking *signets*.



Deep signature transforms

P. Bonnier et al. (2019)

Prior to the truncated signature transform, augment the time series to keep the information that would otherwise be discarded by the truncation

$$x \mapsto \begin{pmatrix} x \\ \phi(x) \end{pmatrix} \mapsto \text{Sig}^N \begin{pmatrix} x \\ \phi(x) \end{pmatrix}.$$

This may be very similar to wavelet or Fourier features.

Nonstationary temporal matrix factorization for multivariate time series forecasting

X. Chen et al.

Temporal matrix factorization reconstructs a multivariate time series from (contemporaneous) latent factors, with a penalty to make them close to VAR.

$$\begin{aligned} \text{Find} \quad & W, X, A_1, \dots, A_d \\ \text{To minimize} \quad & \frac{1}{2} \|P_\Omega(Y - W'X)\|_F^2 \\ & + \frac{1}{2} \rho(\|W\|_F^2 + \|X\|_F^2) \\ & + \frac{1}{2} \lambda \sum_t \|x_t - \sum_k A_k x_{t-k}\|_2^2 \end{aligned}$$

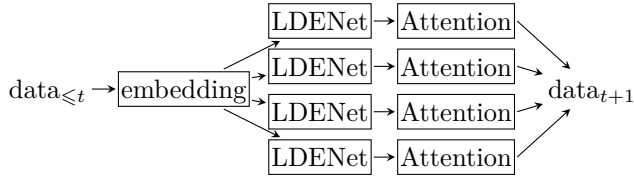
If the time series are not stationary, differentiate them first.

**LDE-Net: Lévy induced
stochastic differential equation
equipped with neural network
for time series forecasting**
L. Yang et al.

Model time series as

$$\begin{aligned} X_{k+1} &= X_k + f(X_k)\Delta t + g(X_k)(\Delta t)^{1/\alpha} L_k \\ L_k &\sim \text{Stable}(\alpha) \\ f, g &: \text{neural nets.} \end{aligned}$$

Combine with an embedding and an attention layer.



**On Hawkes processes
with infinite mean intensity**
C. Aubrun et al. (2022)

The linear Hawkes process has intensity

$$\lambda_t = \lambda_\infty + \int_{-\infty}^t \phi(t-u) dN_u.$$

It can be generalized to

$$\begin{aligned} \lambda_t &= \lambda_\infty + \int_{-\infty}^t \phi(t-u) dN_u + \\ &\int_{-\infty}^t \int_{-\infty}^t Q(t-s, t-u) \varepsilon_u \varepsilon_s dN_u dN_s \end{aligned}$$

(with $\varepsilon_t = \pm 1$). For instance, the ZHawkes model uses

$$Q(s, u) = \phi(s)\delta(s-u) + z(s)z(u).$$

**Distributionally robust risk evaluation with
causality constraint and structural information**
B. Han (2022)

Optimal transport (OT) computes the Wasserstein distance

$$W(\mu, \nu) = \inf_{\pi \in \Pi(\mu, \nu)} \int_{X \times Y} c(x, y) \pi(dx, dy).$$

For distributions on time series, *causal optimal transport* (COT) only considers causal transport plans: $\pi(y_t|x_{1:T}) = \pi(y_t|x_{1:t})$ (it can be formulated as a linear constraint). A risk measure ρ can be made distributionally robust by taking the worst case in a Wasserstein ball around the empirical distribution

$$\max_{W(\mu, \hat{\mu}) \leq \varepsilon} \rho(\mu).$$

**What affects the relationship
between oil prices and the US stock market?**
A mixed-data sampling copula approach

The *copula-Midas-X* model models a high-frequency (daily) bivariate time series (u, v) using low frequency (monthly) time series X_s .

$$\begin{aligned} (u_t, v_t) &\sim C(\lambda_t) && \text{copula} \\ \lambda_t &= \lambda_\tau + \alpha \Phi^{-1}(u_t) \Phi^{-1}(v_t) + \beta \lambda_{t-1} && \text{daily} \\ \lambda_\tau &= \sum_s \gamma_s \sum_k \phi_k(w_s) X_{s, \tau-k} && \text{monthly} \end{aligned}$$

γ_s : weight of predictor s

$\phi_k(w_s)$: weight decay of predictor s

$$\phi_k(w) \propto \left(1 - \frac{k}{K}\right)^{w-1}, \quad \phi_k(w) = 1$$

Try with the symmetrized Joe-Clayton (SJC) copula.

**Structural clustering of volatility regimes
for dynamic trading strategies**
A. Prakash et al. (2021)

To determine the number of volatility regimes (in a return time series):

- Partition the time series into stationary segments (Mood test, in R's `cpm` package to detect changes in variance)
- Compute the Wasserstein distance matrix

$$D_{ij} = W_p(F_i, F_j) = \left(\int |F_1 - F_j|^p \right)^{1/p}$$

- Use spectral clustering, with affinity matrix

$$A_{ij} = \exp - \frac{D_{ij}^2}{\sigma_i \sigma_j},$$

where σ_i is the distance between i and its $\lceil \sqrt{m} \rceil$ th nearest neighbour, where m is the number of segments.

The resulting strategy could be: buy the S&P when volatility is low, switch to gold when it is high.

To compare (x_i, \dots, x_s) and (x_1, \dots, x_t) , replace the observations by their ranks and compare $\langle (x_i - \bar{x})^2 \rangle$ in each segment.

**High-frequency-based volatility model
with network structure**
H. Yuan et al.

The HEAVY volatility model combines high and low frequency data.

$$\begin{aligned} r_t &: \text{daily returns} \\ \text{RM}_y &: \text{daily realized volatility} \\ h_t &= \text{Var}_{t-1}[r_t] \\ \mu_t &= \text{E}_{t-1}[\text{RM}_t] \\ h_t &\sim h_{t-1} + \text{RM}_{t-1} \\ \mu_t &\sim \mu_{t-1} + \text{RM}_{t-1} \end{aligned}$$

It can be generalized to N assets ($N = 18$, in the example), linked through an adjacency matrix A (e.g., industry membership).

$$h_{i,t} \sim h_{i,t-1} + \text{RM}_{i,t-1} + \frac{1}{d_i} \sum_j a_{ij} \text{RM}_{j,t-1}$$

$$\mu_{i,t} \sim \mu_{i,t-1} + \text{RM}_{i,t-1} + \frac{1}{d_i} \sum_j a_{ij} \text{RM}_{j,t-1}$$

***Learning probability distributions
in macroeconomics and finance***
J. Baruník and L. Hanus (2022)

Forecast the h -period ahead distribution of GDP growth, inflation, unemployment from 60 years of 200 macroeconomic variables (FREQ-QD), with a multi-output neural net (MLP, RNN, LSTM) forecasting the binary variables $[y_{t+h} \geq q_t^\alpha]$, where q_t^α is the α -quantile of y on $[0, t]$, for 20 values of α (0.01 to 0.99), with a penalty to ensure the quantiles are in the correct order.

Implementation in Julia (Flux.jl): DistrNN.jl.

***Complexity and persistence of price time
series of the European electricity spot market***
C. Han et al.

To look at the tails of a distribution, plot its (kernel) density with a logarithmic vertical scale and fit a symmetric Lévy α -stable distribution

$$L(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \phi(t) e^{-ixt} dt$$

$$\phi(t) = \exp[it\mu - |ct|^\alpha] \quad \alpha < 2$$

or a q -Gaussian distribution

$$G(x) \propto e_q(-c(x - \mu)^2)$$

$$e_q(x) = [1 - (1 - q)]^{1/(1-q)} \quad q > 1$$

using $\text{KL}(\text{data}||p)$.

To show multifractality, plot the return variance versus the window size: if the Hurst exponent is constant, it is a straight line.

***Machine learning forecast disagreement
and equity returns***
T.G. Bali et al. (2022)

Dispersion between ML models trained on the same inputs have some predictive power on future returns.

***Gamma and vega hedging using
deep distributional reinforcement learning***
J. Cao et al. (2022)

Assuming that Δ -hedging (of an option portfolio) can be done without cost, use D3PG to learn a gamma and vega hedging strategy using options, where the state is

(asset price, portfolio gamma, portfolio vega), the action is the number of options to hold, and the reward is

change in portfolio value – option cost.

***Always safe: reinforcement learning without
safety constraint violations during training***
T.D. Simão et al. (2021)

Factored constrained Markov decision processes (CMDP) use a small subset of the features to describe the dynamics relevant to the safety constraints: the safety dynamics are easier to learn, and the model can be deployed, safely, to continue learning.

***Constrained optimal stopping time
under a regime-switching model***
T. Arai and M. Takenaka (2022)

The optimal stopping time

$$\text{Argmax}_{\tau} \mathbb{E}[e^{-r\tau} \pi(X_{\tau}) \mid X_0 = x]$$

$$\pi(x) = \mathbb{E}\left[\int_t^{\infty} e^{-r(t-s)} X_s ds - I \mid X_t = x\right]$$

is sometimes of *threshold type*

$$\tau^* = \inf\{t > 0 : X_t \geq x^*\}$$

(X_t is a regime switching geometric Brownian motion, and the stopping time is constrained to exogenous random times (Poisson) and regime 1).

***textnets: a Python package
for text analysis with networks***
J.D. Boy

Community detection, on a term-document bipartite graph, is an alternative to LDA topic models.

***Combining natural language processing
and network analysis to examine
how advocacy organizations stimulate
conversation on social media***
C.A. Bail

To get more reactions, on social media, blend unrelated topics (e.g., autism and vaccines): advocacy organization with a high *betweenness centrality* (in a graph built from term co-occurrences), *i.e.*, “cultural bridges”, lead to more reactions.

***Landscape of academic finance
with the structural topic model***
D. Ardia et al. (2022)

The *structural topic model* (STM) extends the latent Dirichlet allocation (LDA) and correlated topic models (CTM) by allowing for document covariates (e.g.,

time).

d : document
 k : topic
 n : word
 X_d : covariates
 θ_d : distribution of topics in document d
 $\theta_d | X_d \sim \text{LogisticNormal}$
 β_k : distribution of words in topic k
 $\beta_k \propto \exp(m + \kappa_k)$
 $z_{d,n} | \theta_d \sim \text{Multinomial}$ topic of word n
 $w_{d,n} | \beta_{z_{d,n}} \sim \text{Multinomial}$ word n

***Do word embeddings really understand
Loughran-McDonald polarities?***
M. Li and C.A. Lehalle (2021)

Word embeddings are bad at sentiment extraction: antonyms tend to be used in the same context (“this was a good film”, “this was a bad film”) – they are frequentist synonyms.

***Heterogeneous
information network based default analysis
on banking micro and small enterprise users***
Z. Zhang et al.

Build a network of industries, small enterprises and persons accounting for:

- Fund transfer between companies
- Person/company controlling/investing in a company
- Industry membership; up/down-stream industries

***Stock embeddings: learning distributed
representations for financial assets***
R. Dolphin et al.

Use word2vec to compute stock embeddings, linking a stock to a context (the k stocks with the closest returns on a random n -day window, $k = 3$, $n = 1$)

Understanding the quintile portfolio
R. Zhou and D.P. Palomar (2020)

The long-only quintile portfolio (more generally, the top $k\%$ portfolio) is the solution of the robust optimization problem

$$\begin{aligned}
 &\text{Find} && w \\
 &\text{To maximize} && \text{Min}_{\mu: \|\mu - \mu_0\|_1 \leq \varepsilon} w' \mu \\
 &\text{Such that} && w \geq 0 \\
 &&& w' \mathbf{1} = 1
 \end{aligned}$$

(for some value of ε); for larger values of ε , it is the $1/N$ portfolio.

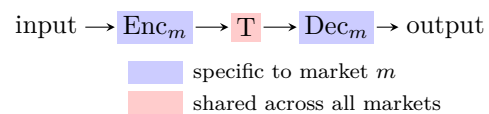
For the long-short quintile portfolio, replace the constraints with $\|w\|_1 = 1$, $w' \mathbf{1} = 0$.

For inverse-volatility weights, replace the uncertainty set with $\{\mu_0 + e : \|e \oslash \sigma\|_1 \leq \varepsilon\}$ where \oslash is the elementwise division.

The results generalize to market exposure $w' \beta = 0$ or sector exposures.

***QuantNet: transferring learning
across trading strategies***
A. Koshiyama et al.

Train 60 country-specific models, with a shared central layer, to output a trading signal in $[-1, 1]$ (weights), maximizing the Sharpe ratio; encoder and decoder are LSTMs.



***The virtue of complexity
in machine learning portfolios***
B. Kelly et al. (2021)

Financial models exhibit the double descent phenomenon.

***Dynamic portfolio optimization
with inverse covariance clustering***
Y. Wang and T. Aste (2022)

Define market states by clustering stock returns, using:

- The Euclidean distance;
- Or the Gaussian or Student log-likelihood, computed using a *sparse* estimate of the inverse covariance matrix.

A factor model for option returns
M. Büchner and B. Kelly

Instrumented PCA allows the loadings to depend on exogenous variables (e.g., moneyness, time to maturity, implied volatility, embedded leverage, Greeks).

$$\begin{aligned}
 r_{it} &: \text{asset returns} \\
 z_{it} &: \text{exogenous variables} \\
 \beta_{it} &= z'_{it} \Gamma + \text{noise} \\
 r_{i,t+1} &= \beta_{it} f_{t+1} + \text{noise} \\
 f_t &: \text{factor returns}
 \end{aligned}$$

For options on the S&P 500 (monthly holding period, delta-hedged daily), 3 factors are needed; they can be interpreted as volatility surface level, term structure slope of the volatility surface, moneyness skew of the volatility surface.

Drawdown beta and portfolio optimization

R. Ding and S. Uryasev (2021)

The *drawdown at risk* (DaR) is the α quantile of the distribution of drawdowns, The *conditional drawdown at risk* (CDaR) is the average of the drawdowns beyond the DaR. The *expected regret of drawdown* (ERoD) is the average of the drawdowns beyond some fixed threshold (instead of a quantile). The *CDaR beta* (resp., ERoD beta) is the average asset return, divided by the market CDaR (resp. ERoD), with the averages in the numerator and denominator are computed over matching periods.

You want to invest in stocks with $\beta \approx 0$ but $\beta_{\text{CDaR}} < 0$.

Capital asset pricing model (CAPM) with drawdown measure

M. Zabarankin et al. (2013)

In the CAPM, replace the 1-period returns with returns over market drawdown periods.

Sensitivity to large losses and ρ -arbitrage for convex risk measures

M. Herdegen and N. Khan (2022)

A risk measure ρ is susceptible to ρ -arbitrage if it is possible to have arbitrary high returns with arbitrarily low risk. If ρ is

- Monotone: $X \leq Y$ a.s. $\implies \rho(X) \geq \rho(Y)$,
- Normalized: $\rho(0) = 0$,
- Star-shaped: $\forall X \forall \lambda \geq 1 \rho(\lambda X) \geq \lambda \rho(X)$,
- Sensitive to large losses:

$$P[X < 0] > 0 \implies \exists \lambda > 0 \rho(\lambda X) > 0,$$

there is no ρ -arbitrage.

Universal approximation of credit portfolio losses using restricted Boltzmann machines

G. Genovese et al. (2022)

Probability equivalent level of value at risk and higher order expected shortfalls

M. Barczy et al.

The *probability equivalent level of VaR and ES*, $\text{PELVE}_\alpha X$ is the $c \geq 1$ such that $\text{VaR}_{1-\alpha} X = \text{ES}_{1-c\alpha} X$. It is motivated by the Basel move from $\text{VaR}_{.99}$ to $\text{ES}_{.975}$ ($c=2.5$). The PELVE can be generalized from (VaR, ES) to $(\rho, \tilde{\rho})$, for families of risk measures $(\rho_p)_p$ and

$$\tilde{\rho}_p = \frac{1}{1-p} \int_p^1 \rho_s(X) ds.$$

The n th order expected shortfall is

$$\text{ES}_{p,n} = \frac{n}{1-p} \int_p^1 \left(\frac{s-p}{1-p} \right)^{n-1} \text{VaR}_s(X) ds.$$

A simple method for measuring inequality

T. Sitthiyot and K. Holasut (2020)

Instead of using only the Gini index, look at several measures:

- Gini index;
- Income share held by the top 10%: T_{10} ;
- Income share held by the bottom 10%: B_{10} (or their ratio, B_{10}/T_{10});
- Entropy H of the income distribution.

Combine those measures as

$$\sqrt{\frac{\text{Gini}^2 + H^2}{2}} \text{ or } \sqrt{\frac{\text{Gini}^2 + \sqrt{1 - \frac{B_{10}}{T_{10}}}}{2}}$$

since $H \approx (1 - B_{10}/T_{10})^{1/4}$.

Missing financial data

S. Bryzgalova et al.

To fill in missing data in panel data $C_{it\ell}$

i : asset
 ℓ : feature
 t : time

estimate a cross-sectional factor model (PCA with missing values, for each t , but with $\Lambda_t = \Lambda$ constant)

$$C_t = F_t \Lambda + \text{noise}$$

$I \times L \quad I \times K \quad K \times L$

and use it in a time series model

$$C_{it} \leftarrow \text{coalesce}(C_{it}, f(C_{i,t-1}, F_{it}))$$

(or $f(C_{i,t-1}, C_{i,t+1}, F_{it})$ if you are willing to use future data) where f is linear.

Vulnerability-CoVaR: investigating the crypto market

M. Waltz et al. (2022)

There are many generalizations of value at risk (VaR) to several assets:

- $\text{CoVaR}^{j|i}$: α quantile of X_j conditioned on X_i being below its α quantile;
- MCoVaR^j : α quantile of X_j conditioned on all the X_i being below their α quantile;
- VCVaR^j : α quantile of X_j conditioned on at least one X_i being below its α quantile;
- $\text{SCoVaR}^{j|i}$: α quantile of X_j conditioned on $\sum_{i \neq j} X_i$ being below its α quantile;
- Vulnerability index (VI): probability that X_j is below its α quantile given that at least one X_i is below its α quantile;
- ΔCoVaR : difference between $\text{CoVaR}^{j|i}$ and the α quantile of X_i being at its 50% quantile.

Evolutionary correlation, regime switching, spectral dynamics and optimal trading strategies for cryptocurrencies and equities
N. James (2022)

Compute the correlation matrix between 45 cryptocurrencies, between 100 equities, and within each sector (for cryptocurrencies, use the `coinmarketcap.com` classification: centralized exchanges, collectibles (NFT), decentralized finance, platform, smart contracts, store of value, wallet – some are in several categories), apply random matrix theory, and plot, over time:

- The number of non-random eigenvalues;
- The largest eigenvalue;
- The proportion of variance explained by the first eigenvalue.

Common risk factors in cryptocurrency
Y. Liu et al. (2019)

Among Fama-French-like factors for crypto currencies (capitalization, age, volume, β , β^2 , volatility, idiosyncratic volatility, skewness, maximum return in the previous week, momentum for various lookback periods, Amihud), market, size and momentum explain cross-sectional returns.

Sequence-based target coin prediction for cryptocurrency pump-and-dump
S. Hu et al.

Detect pump and dump announcements from 1000 Telegram channels (start on PumpOlymp and follow the invitation links; use tf-idf vectors) and try to predict the target coin.

A word is worth a thousand dollars: adversarial attacks on Tweet fools stock prediction
Y. Xie et al.

Retweet semantically similar adversarial tweets (replace a single word) to attack Stocknet, FinGRU and FinLSTM.

NFT appraisal prediction: utilizing search trends, public market data, linear regression and recurrent neural networks
S. Jain et al.

Forecast NFT prices using:

- The degree and PageRank centrality of buyers and sellers; user activity on Twitter and OpenSea;
- The first 5 principal components of AlexNet embeddings of the NFT;
- Price history (in the NFT collection, using the `covalent` API); gas (ETH transaction fee);
- BTC, ETH, Gold, S&P 500, Nasdaq 100, FAANG;
- Google search volume (`pytrends`); news sentiment.

Constructing and NFT index and applications
H. Schnoering and H. Inzirillo

Build an NFT index from 60 collections:

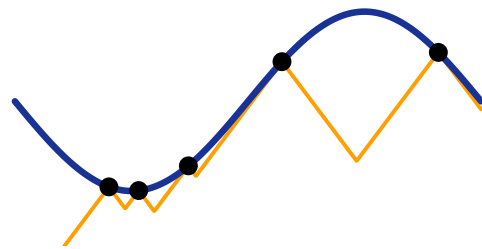
$$\log(\text{price}_{ijt}) \sim \text{collection}_j + \text{scarcity}_{ij} + \log(\text{index}_t) + \text{noise}_{it}$$

where scarcity_{ij} measures the scarcity of the traits of NFT i in collection j .

Algorithms for optimization
M.J. Kochenderfer and T.A. Wheeler (2019)

3. Given a univariate function f , three points $a < b < c$ are said to **bracket** a minimum if $f(b) < f(a)$ and $f(b) < f(c)$. *Fibonacci search* splits an interval in an optimal way (to obtain the smallest bracket after n iterations); it is often approximated with a *golden section search*. *Quadratic fit search* (often faster) improves a bracket by approximating the function with a quadratic and taking its minimum.

For a Lipschitz continuous function, the *Schubert-Piyavskii method* keeps track of a lower bound on the function and progressively improves it.



If the derivative of the objective f is known, the bisection method can solve $f'(x) = 0$.

4. *Line search* starts with a point x , a direction d , and solves the 1-dimensional problem

$$\underset{\alpha}{\text{Minimize}} f(x + \alpha d).$$

Approximate line search looks for a step size providing a “sufficient decrease” (Armijo condition)

$$f(x + \alpha d) \leq f(x) + \beta \alpha \nabla_d f(x) \quad \beta = 10^{-4}.$$

The Wolfe conditions also include

$$\nabla_d f(x + \alpha d) \geq \sigma \nabla_d f(x).$$

Trust region methods forbid excessively large step sizes

$$\underset{x}{\text{Minimize}} \hat{f}(x) \text{ such that } \|x - x_0\| \leq \delta;$$

the radius δ can be adjusted by looking at the local model performance

$$\eta = \frac{\text{actual improvement}}{\text{predicted improvement}} = \frac{f(x) - f(x_0)}{\hat{f}(x) - f(x_0)}.$$

5. Gradient descent can perform poorly in narrow valleys. *Conjugate gradient* minimizes a quadratic function $f(x) = \frac{1}{2}x'Ax + b'x + c$ in n mutually conjugate

steps $\forall i \neq j \ d'_i A d_j = 0$.

$$d_k = -g_k + \beta_k d_{k-1}$$

$$\beta_k = \frac{g'_k A d_{k-1}}{d'_{k-1} A d_{k-1}}$$

We do not know A , but we can use approximations.

$$\beta_k = \frac{g'_k g_k}{g'_{k-1} g_{k-1}} \text{ or } \beta_k = \frac{g'_k (g_k - g_{k-1})}{g'_{k-1} g_{k-1}}$$

Gradient descent can be improved with:

- Momentum
- Nesterov momentum (momentum at the next point)
- An adaptive learning rate for each coordinate, using $\sum_k g_{ki}^2$ (AdaGrad, but the learning rate decreases too much) or an exponential moving average of the g_{ki}^2 (RMSProp, which has inconsistent units, or AdaDelta);
- Adam, which combines momentum and adaptive learning rate.

Hypergradient descent also optimizes the learning rate.

6. *Newton's method* approximates the objective function with a quadratic function, $x \leftarrow x - H^{-1}g$; it can fail (oscillations, overshoot, or $f'' < 0$). The *secant method* uses an approximation of the second derivative with the first (in dimension 1). *Quasi-Newton methods* use an approximation of the Hessian (DFP, BFGS, L-BFGS).

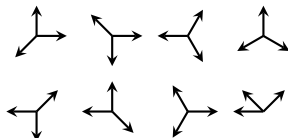
7. *Coordinate descent* (aka cyclic coordinate search) optimizes one coordinate at a time. *Powell's method* is similar, but progressively changes the directions, replacing the last one with $x_{n+1} - x_1$ (occasionally reset the search direction lest they become colinear).

The *Hooke-Jeeves* method evaluates $f(x)$ and $f(x \pm \alpha e_i)$. *Generalized pattern search* is similar but evaluates $f(x)$ and $f(x + \alpha d)$, $d \in \mathcal{D}$, where \mathcal{D} is a *positive spanning set*, which can have as few as $n + 1$ elements.

The *Nelder-Mead method* uses a simplex to explore the space.

The *divided rectangle* (DIRECT) method uses Lipschitz bounds, but without specifying the Lipschitz constant (it considers all possible values), and dividing the search space recursively, into rectangles (otherwise, the Lipschitz lower bound is a (rather complex) intersection of cones); rectangles for which there is a Lipschitz constant for which their lower bound would be a minimum are split into thirds along their largest dimension.

8. To avoid plateaus and saddle points, try noisy descent (adding Gaussian noise to the gradient), *mesh adaptive direct search* (random positive spanning directions)



or, even, simulated annealing.

The *cross-entropy method* does not follow a point but a distribution; at each step, it samples from that distribution, and fits a distribution to the best samples.

Natural evolution strategies also solve

$$\underset{\theta}{\text{Minimize}} \quad \mathbb{E}_{x \sim p_\theta} [f(x)]$$

but uses gradient descent and the log-derivative trick.

CMA-ES differs from CEM:

- After estimating μ, σ , it samples from $N(\mu, \sigma^2 \Sigma)$;
- Instead of selecting the best samples, it weighs them (some weights are negative)

$$w_i = \log \frac{m+1}{2} - \log i \quad i \in \llbracket 1, m \rrbracket;$$

- The step size is updated (in a complicated way).

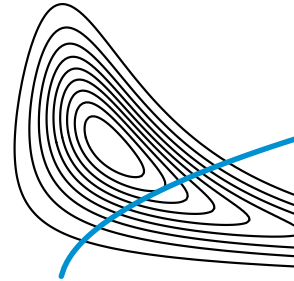
9. Initialize population methods with a uniform, Gaussian or Cauchy: genetic algorithms (cross-over and mutations), *differential evolution*

$$x = x_i + w(x_j - x_k),$$

particle swarm optimization (momentum), firefly algorithm (the attractive power of each particle depends on its fitness), etc. – there is an excessive proliferation of nature-inspired methods.

Hybrid methods combine population search with local search.

10. Transforming a constrained optimization problem can sometimes help remove constraints.



Lagrange multipliers deal with equality constraints

$$\underset{x}{\text{Minimize}} \quad f(x) \text{ such that } h(x) = 0.$$

The first-order conditions

$$h(x) = 0$$

$$\nabla f(x) = \lambda \nabla h(x)$$

can be written with the Lagrangian

$$\mathcal{L}(x, \lambda) = f(x) - \lambda h(x)$$

$$\nabla_x \mathcal{L} = 0$$

$$\nabla_\lambda \mathcal{L} = 0.$$

With inequality constraints,

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & f(x) \\ \text{Such that} & g(x) \leq 0 \\ & h(x) = 0 \end{array}$$

things are more complicated. The *KKT conditions* are

$$\begin{aligned} g &\leq 0 \\ h &= 0 \\ \mu &\geq 0 \\ \mu \odot g &= 0 \\ \nabla f + \mu' \nabla g + \lambda' h(x) &= 0. \end{aligned}$$

With the generalized Lagrangian

$$\mathcal{L}(x, \mu, \lambda) = f(x) + \mu' g(x) + \lambda' h(x),$$

the primal and dual problems are

$$\begin{aligned} \text{primal} \quad & \text{Minimize} \text{Max}_{x, \mu \geq 0, \lambda} \mathcal{L}(x, \mu, \lambda) \\ \text{dual} \quad & \text{Maximize} \text{Min}_{x, \mu \geq 0, \lambda} \mathcal{L}(x, \mu, \lambda). \end{aligned}$$

The max-min inequality is

$$\text{Max}_a \text{Min}_b f(a, b) \leq \text{Min}_b \text{Max}_a f(a, b).$$

To solve constrained optimization problems, try adding a quadratic penalty

$$\sum_i \text{Max}(g_i(x), 0)^2 + \sum_j h_j(x)^2.$$

For equality-constrained problems, the *augmented Lagrange method* uses both a quadratic and a linear penalty

$$\begin{aligned} p(x) &= \frac{1}{2} \rho \|h(x)\|^2 - \lambda' h(x) \\ \lambda_{k+1} &= \lambda_k - \rho h(x). \end{aligned}$$

Interior point methods use a *barrier penalty* (infinite for infeasible points), e.g.,

$$p(x) = -\sum \frac{1}{g_i(x)} \text{ or } p(x) = -\sum \log(-g_i(x)) \mathbf{1}_{g_i(x) \geq -1}$$

(to find a feasible point, use a quadratic penalty).

11. The *simplex algorithm* moves from vertex to vertex of the feasible set.

To find a feasible solution of

$$\text{Minimize}_{x, z} c'x \text{ such that } Ax = b, x \geq 0,$$

add extra variables z and try to zero them out

$$\begin{aligned} \text{Find} \quad & x, z \\ \text{To minimize} \quad & \begin{pmatrix} 0 \\ 1 \end{pmatrix}' \begin{pmatrix} x \\ z \end{pmatrix} \\ \text{Such that} \quad & (A \quad Z) \begin{pmatrix} x \\ z \end{pmatrix} = b \\ & x, z \geq 0 \end{aligned}$$

where $Z = \text{diag}(\text{sign}(b))$.

12. To solve multiobjective optimization problems:

- Constrain all but one of the objectives;
- Rank the objectives, and maximize them one by one, adding constraints to fix the previous ones;

or combine the objectives:

- $w' f(x)$
- $\|f(x) - y_{\text{goal}}\|_p$ (goal programming)
- $\sum w_i (f_i(x) - y_i^{\text{goal}})^p$
- $\text{Max}_i w_i (f_i(x) - y_i^{\text{goal}})^p$
- $\sum (e^{pw_i} - 1) e^{pf_i(x)}$.

NSGA-II is a population method keeping track of the level of the individuals (level-1 are non-dominated, level $k+1$ are non-dominated after removing levels 1 to k). To preserve diversity, reinject good points into the population, or penalize points with the density of their neighbourhood.

To choose w in the weighted model $w' f(x)$, ask experts if they prefer a or b , and pick w such that $w'a < w'b$; repeat with other pairs; prefer the w that best separates $w'a$ and $w'b$. Choose a and b to cut the feasible space \mathcal{W} into two equal parts (Q-eval).

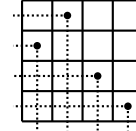
Robust optimization does not choose one $w \in \mathcal{W}$:

$$\text{Minimize}_x \text{Max}_{w \in \mathcal{W}} w' f(x).$$

The *minimax regret* approach solves

$$\text{Minimize}_x \text{Max}_{w \in \mathcal{W}} \text{Max}_y w' f(x) - w' f(y).$$

13. Grid search (full factorial design) does not scale to high dimensions: try random sampling. A *uniform projection plan* is a sampling plan over a grid with uniform margins (latin hypercube sampling).



To measure how space-filling a sampling plan is, look

$$\text{discrepancy}(X) = \sup_{H \text{ hyperrectangle}} \left| \frac{\#X \cap H}{\#X} - \text{vol}(H) \right|$$

$$d(X) = \text{Min}_{i \neq j} d(x_i, x_j)$$

$$\Phi_q(X) = \left(\sum_{i \neq j} d(x_i, x_j)^{-q} \right)^{1/q}$$

$$\Phi(X) = \text{Max}_{q \in \{1, 2, 5, 10, 20, 50, 100\}} \Phi_q(X)$$

Quasi-Monte Carlo (with quasi-random sequences, such as additive recurrences, Halton or Sobol sequences) has an error $O(1/m)$ instead of $O(1/\sqrt{m})$.

14. One can build a *surrogate model* with basis functions (sines, polynomials, radial basis functions) and assess it with cross-validation or *0.632 bootstrap* (linear combination of leave-out-one bootstrap and bootstrap, to avoid bias).

15. Gaussian processes (GP) are probabilistic surrogate models.

16. To choose where next to evaluate the objective function, look at the uncertainty $\hat{\sigma}(x)$, a lower confidence bound $\hat{\mu}(x) - \alpha\hat{\sigma}(x)$, the probability of improvement, or the expected improvement.

To ensure safe exploration, add a “safety score” to the GP model.

17. To deal with noise, e.g., $f(x, z) = f(x) + z$ or $f(x, z) = f(x + z)$, with an uncertainty set, use the minimax

$$\underset{x}{\text{Minimize}} \quad \underset{\|z\| \leq \varepsilon}{\text{Max}} \quad f(x, z)$$

or the information gap

$$\begin{array}{ll} \text{Find} & x, \varepsilon \\ \text{To maximize} & \varepsilon \\ \text{Such that} & \forall z \quad \|z\| \leq \varepsilon \Rightarrow (x, z) \text{ feasible} \\ \text{and} & f(x, z) \leq y_{\max}. \end{array}$$

Probabilistic uncertainty uses distributions over the noise z : look at $\mathbb{E}_z f(x, z)$, $\text{Var}_z f(x, z)$, $\text{VaR}_z^{1-\alpha} f(x, z)$, $\text{CVaR}_z^{1-\alpha} f(x, z)$.

18. Given a distribution $z \sim p$, we can estimate the distribution of $f(z)$ (in particular $\mathbb{E} f(z)$ and $\text{Var} f(z)$) using:

- A Taylor expansion of f ;
- Monte Carlo samples;
- Polynomial chaos: if f is a polynomial (product of orthogonal polynomials), we can easily compute $\mathbb{E} f(z)$ and $\text{Var} f(z)$;
- Gaussian processes: $\mathbb{E}_z f(z) \approx \mathbb{E}_{\hat{f}} \mathbb{E}_z \hat{f}(z) = \mathbb{E}_z \mathbb{E}_{\hat{f}} \hat{f}(z) = \mathbb{E}_z \mu(z)$ (it is a little more complicated for $\text{Var} f(z)$).

19. For an integer program

$$\underset{x}{\text{Minimize}} \quad c'x \text{ such that } Ax = b, \quad x \geq 0$$

with a *totally unimodular* matrix A (the determinant of all submatrices is 0, +1 or −1), the simplex algorithm returns an integer solution.

The *cutting plane* method solves the problem without the integrality constraint, and adds constraints to remove the non-integral parts of the solution. *Branch and bound* recursively splits the problem by adding constraints: $x_i \leq \lfloor x_i^* \rfloor$ and $x_i \geq \lceil x_i^* \rceil$.

20. To explore a search space described by a grammar, try genetic programming (cross-over swaps two random subtrees). A probabilistic grammar, with probabilities progressively updated to focus on derivation rules improving the objective, may help.

21. Multi-disciplinary optimization studies problems of the form

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & f(x) \\ \text{Such that} & \forall i \quad F_i(x) = x \end{array}$$

where F_i only modifies coordinate i of x . Try (cf pro-

jected gradient)

$$\begin{array}{l} x \leftarrow \text{MDA}(x) \\ x \leftarrow \text{improve}(x) \end{array}$$

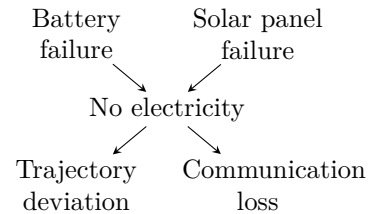
where MDA returns a compatible assignment.

Algorithms for decision making M.J. Kochenderfer et al. (2022)

1. Decision making algorithms can rely on:

- Explicit programming (hard-coded policies);
- Supervised learning (behavioural cloning);
- Optimization (search in policy space);
- Planning (reduction of the problem to dynamic programming);
- Reinforcement learning (if the model of the environment is not known).

2. Probability distributions on a large number of variables can be represented by Bayesian networks

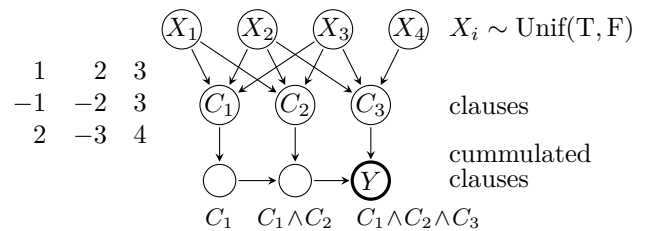


explaining away: $B \not\perp\!\!\!\perp S \mid E$

3. Inference on Bayesian networks can use:

- Sum-product variable elimination (marginalizing out variables as early as possible);
- Junction tree algorithm (belief propagation on a tree of subsets of variables);
- Loopy belief propagation.

Exact inference is NP-hard: a 3-SAT problem can be converted to a Bayesian network.



Approximate inference can also be done via:

- Direct sampling (use a topological order on the nodes);
- Likelihood-weighted sampling;
- Gibbs sampling.

4. The parameters of a Bayesian network can be learned via

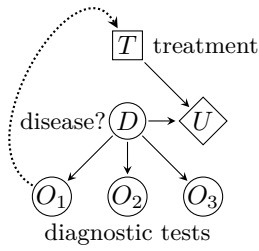
- Maximum likelihood;
- Bayesian methods (with beta or Dirichlet priors);
- Expectation maximization (EM), if some of the data is missing.

5. To learn the structure of a Bayesian network, compute the score (penalized log-likelihood) of candidates, using local search or *K2 search* (iterate over the nodes in a prespecified order, adding parents if this increases the score, with a limit on the number of parents). It may be more efficient to search the space of essential graphs (CPDAG) instead of all DAGs.

6. Under some reasonable assumptions on an agent's preferences, rational decisions are those maximizing the expected utility, for some utility function.

A *decision network* (influence diagram) has

- Chance, decision and utility nodes;
- Conditional, informational and functional nodes (ending in chance, decision, and utility nodes).

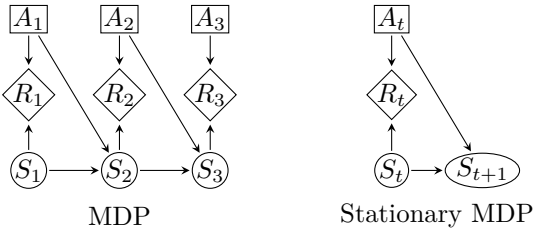


The *value of information* is

$$\text{VOI}(O_2|O_1) = \mathbb{E}_{O_2} [\text{EU}(O_1, O_2)] - \text{EU}(O_1)$$

where $\text{EU}(O)$ is the expected utility of an optimal decision given observation O .

7. Many problems require a sequence of decisions; with a Markov assumption, they can be modeled as an MDP.



Policy evaluation computes the value (expected return) of a state (if we follow the policy π) by iterating

$$U(s) \leftarrow R + \gamma \mathbb{E}_{s'} [U(s')].$$

Policy iteration computes an optimal policy by iterating between policy evaluation and policy improvement (replacing the current policy with the greedy policy wrt the value function).

Value iteration computes an optimal policy by iterating Bellman's equation

$$U(s) \leftarrow \text{Max}_a (R(s, a) + \gamma \mathbb{E}_{s'} [U(s')]);$$

the updates can be done asynchronously.

The optimal policy is also solution of the linear program

$$\begin{aligned} &\text{Find} && U \\ &\text{To minimize} && \sum_s U(s) \\ &\text{Such that} && \forall s \ U(s) \geq \text{Max}_a (R(s, a) + \gamma \mathbb{E}_{s'} [U(s')]) \end{aligned}$$

(the minimization turns the inequalities into equalities).

8. If the state space is large (or continuous), approximate the value function: k -nearest neighbour, kernel smoothing, linear interpolation, simplex interpolation (only uses $d + 1$ neighbours in dimension d , instead of 2^d for linear interpolation), linear regression, neural networks. Many of those can be written $U_\theta(s) = \theta' \beta(s)$ for some basis functions $(\beta_i)_{1 \leq i \leq m}$.

9. The reachable state space is often smaller than the full state space: *receding horizon planning* solves a subset of the full MDP by only looking d steps ahead.

- Expand the whole search tree (forward search);
- It may be possible to prune the search tree using a lower bound on the state value function $U(s)$ and an upper bound on the state action value function $Q(s, a)$;
- Sample the next states using a rollout policy and average;
- Monte Carlo tree search (MCTS) not only samples but also updates estimates of the state action value function $Q(s, a)$, often using the UCB1 exploration heuristic

$$\text{UCB1} = Q(s, a) + c \sqrt{\frac{\log N(s)}{N(s, a)}};$$

limit (but progressively increase) the number of actions considered in a state, and the number of states reachable after an action, to $\theta_1 N(s)^{\theta_2}$ and $\theta_3 N(s, a)^{\theta_4}$;

- Heuristic search samples from a greedy policy for value function U , initialized to an upper bound, and progressively refined.

Openloop planning (committing to the next d actions, i.e., not revising the plan when new observations arrive), reduces to an optimization problem; there are robust variants.



10. *Policy search* directly searches the space of policies to maximize the expected discounted return $U(\theta)$, often approximated using m strategy rollouts.

The *Hooke-Jeeves method* is a local search taking steps of size $\pm \alpha$ in each coordinate direction.

Genetic search keeps track of a population of policies, progressively perturbed and pruned.

The *cross-entropy method* (CEM) updates a distribution rather than a set of points

$$\text{Maximize}_{\psi} \mathbb{E}_{\theta \sim p_{\psi}} U(\theta)$$

where $U(\theta)$ is the expected discounted return under policy π_θ ; the expectation is approximated with samples from p_ψ , and the distribution is updated by fitting it to the best samples.

Evolution strategies optimize the same objective function, by using a gradient step, computed with the *log-derivative trick*

$$\nabla_{\psi} \mathbb{E}_{\theta \sim p_\psi} U(\theta) = \dots = \mathbb{E}_{p_\psi} [U(\theta) \nabla_{\psi} \log p_\psi(\theta)].$$

Rank shaping replace the utilities $U(\theta)$ (which are noisy) with their rank, or some transformation thereof.

11. To estimate the gradient $\nabla U(\theta)$, one could use:

- Finite differences;
- Regression, after estimating $U(\theta + \varepsilon)$ for random offsets ε (instead of fixed-size, axis-aligned offsets);
- The log-derivative trick; the variance can be reduced by using the reward-to-go, and by subtracting a baseline.

12. *Gradient ascent* takes a step in the gradient direction; the gradient can be *clipped* (coordinate-wise) or rescaled (to cap its norm). Rescaling the gradient to set its norm (*restricted gradient*) is equivalent to maximizing a linear approximation of the objective inside a ball; the *natural gradient* defines that ball with the Fisher information matrix; the *trust region method* (TRPO) uses a line search in the gradient direction within the Fisher information ball.

13. *Actor-critic* methods jointly optimize the policy (actor) and refine an approximation of the value function (critic): by providing a better baseline, the critic helps lower the variance of the policy gradient.

14. To evaluate the final policy:

- Use importance sampling, to focus on rare events;
- Use robust dynamic programming, or adversarial analysis;
- Compute the Pareto frontier, if there are several competing performance measures.

15. To balance exploration and exploitation, for binary *bandits* (single-state MDP, with binary rewards) one can take random actions once in a while (ε -greedy).

Optimism under uncertainty kept track of an estimate of the (distribution of the) value of each action, and chooses the action (“arm”) with the highest α -quantile, or some other upper confidence bound, e.g.,

$$\text{UCB}_1(a) = \rho_a + c \sqrt{\frac{\log N}{N(a)}}.$$

Dynamic programming on belief states (number of tries, number of wins, for each arm – there are exponentially many beliefs) can be solved efficiently with the *Gittins allocation index* [no details].

16. The MLE of the MDP can be obtained from the counts, and solved at each step; instead of re-solving it each time, a few Bellman updates may be sufficient,

e.g., focusing on states known to lead to the current state. One can bias the estimated MDP to encourage exploration (increasing the reward of insufficiently-visited states).

Bayes-adaptive MDPs extend the state space S to $S \times B$ where B is the set of possible beliefs on the model parameters.

Posterior sampling (Thompson sampling) samples from the posterior model parameters, instead of using the MAP.

17. Model-free methods model the state action value function Q directly, without estimating the MDP.

$$\text{Bellman} \quad Q(s, a) = \mathbb{E}_{r, s'} [r + \gamma \text{Max}_{a'} Q(s', a')]$$

$$\text{Update} \quad Q(s, a) \leftarrow Q(s, a) + \alpha [\text{target} - Q(s, a)]$$

$$\text{Q-learning} \quad \text{target} = r + \gamma \text{Max}_{a'} Q(s', a') \quad \text{off-policy}$$

$$\text{SARSA} \quad \text{target} = r + Q(s', a') \quad \text{on-policy}$$

With sparse rewards, learning can be slow.

Eligibility traces apply the update

$$\begin{aligned} Q(s, a) &\leftarrow Q(s, a) + \alpha \delta \\ \delta &= \text{target} - Q(s, a) \end{aligned}$$

to all state-action pairs, with an exponentially decaying coefficient λ , increased each time the pair is visited

$$Q(s, a) \leftarrow Q(s, a) + \lambda(s, a) \cdot \alpha \cdot \delta$$

(it may not work well with off-policy updates, such as Q-learning).

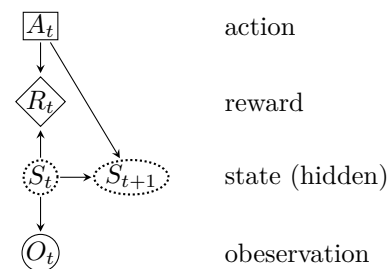
Experience replay can help avoid *catastrophic forgetting*.

18. *Behavioural cloning* uses supervised learning to reproduce expert decisions, but suffers from cascading errors: small errors compound and eventually lead the agent to regions of the state space never visited by the expert.

Inverse reinforcement learning learns a reward function (not a policy) from trajectories, by matching policy features. The problem is underspecified: also maximize the entropy of the distribution over trajectories.

Generative adversarial imitation learning (GAIL) uses a GAN to generate expert-like trajectories.

19. An MDP is a conditional distribution $r, s' | s, a$. A POMDP is a conditional distribution $r, s', o | s, a$.



The *Kalman filter* progressively updates beliefs about the state, assuming the model is linear and Gaussian.

The *extended Kalman filter* (EKF) linearizes non-linear models (with Gaussian noise).

The *unscented Kalman filter* (UKF) is derivative-free, and uses deterministic sampling (“delta points”) to approximate non-linear functions (“unscented transform”).

$$\mu, \Sigma \mapsto s_i^\pm = \mu \pm \sqrt{\Sigma_i} \mapsto f(s_i^\pm) \mapsto \mu', \Sigma'$$

Particle filters (sequential Monte Carlo, SCM) can deal with nonlinear, non-Gaussian models, but can fail because of *particle deprivation*. *Particle injection* tried to remedy this by occasionally injecting particles from a broad distribution (e.g., uniform), when the mean particle weight becomes too high.

20. A POMDP can be seen as an MDP with beliefs as states; for planning, only use the k -step-ahead sub-MDP.

21. *QMDP* updates a set of *alpha vectors*, one for each action.

$$Q(b, a) = \sum_s Q(s, a) b(s) = \alpha'_a b$$

$$\alpha_a(s) = R(s, a) + \gamma \sum_{s'} T(s'|s, a) \text{Max}_{a'} \alpha_{a'}(s')$$

The *fast informed bound* is similar, but accounts for the observation model. For lower bounds, try best action worst state, or single action forever.

Point-based value iteration uses a set of alpha vectors, associated to different belief points, $U(b) = \text{Max}_{\alpha \in \Gamma} \alpha' b$.

22. MCTS can be extended to POMDPs.

23. A *controller* (finite state controller, FSA) is a policy with an internal state.

24. In a *simple game* (single-state, multi-agent MDP), a *dominant strategy* is a best response against all possible agent policies; a *dominant strategy equilibrium* need not exist.

Games with a finite state space have a Nash equilibrium: each agent follows a best-response strategy. Computing a Nash equilibrium is PPAD-complete.

A *correlated equilibrium* is a joint policy (the agent’s actions need not be independent) in which no agent can change its action to increase its expected utility.

Hierarchical softmax models human agents:

- Level 0 agents play at random;
- Level $k + 1$ agents assume the other players are level k , and select actions with a softmax with precision λ .

25. A *Markov game* (MG) is a multi-agent MDP. Policy evaluation, best response, softmax response

$$\pi_i(a_i|s) \propto \exp \lambda Q(s, a_i),$$

Nash equilibrium can be generalized to MGs. To improve the strategy, try gradient ascent with fictitious play, or Nash Q-learning.

26. A partially-observable Markov game (POMG) is a multi-agent POMDP.

27. In a *decentralized POMDP*, agents share the same reward.

Bayesian optimization R. Garnett (2021)

1. *Sequential optimization* iterates

$$x \leftarrow \text{policy}(\mathcal{D})$$

$$y \leftarrow \text{observe}(x)$$

$$\mathcal{D} \leftarrow \mathcal{D} \cup \{(x, y)\}.$$

The new observation, x , is often selected by maximizing some function (*acquisition* function).

2. *Gaussian processes* are distributions on functions whose restrictions on finite sets are Gaussian – they are “very high-dimensional” Gaussian distributions: $f \sim \text{GP}(\mu, k)$ means (writing ϕ instead of $f(\mathbf{x})$, to simplify the notations)

$$\phi \sim N(\mu, \Sigma)$$

$$\mu = \mathbb{E}[\phi] = \mu(\mathbf{x})$$

$$\Sigma = \text{Cov}[\phi] = k(\mathbf{x}, \mathbf{x}).$$

Values are often observed with noise: $\mathbf{y} \sim N(\mu, \Sigma + N)$.

The posterior is a conditional Gaussian:

$$\mu_{\mathcal{D}} = \mu(a) + k(x, \mathbf{x})(\Sigma + N)^{-1}(\mathbf{y} - \mu)$$

$$k_{\mathcal{D}}(x, x') = k(x, x') - kx(x, \mathbf{x})(\Sigma + N)^{-1}k(\mathbf{x}, x').$$

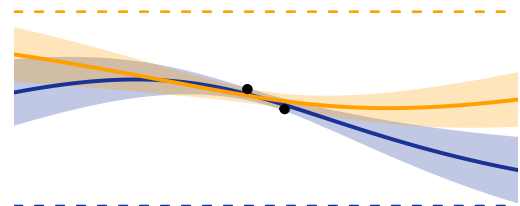
If L is linear and X Gaussian. then (X, LX) is Gaussian. Since ∇ or $h = \text{vech} \nabla \nabla^\top$ are linear, if f is a GP, then so if $(f, \nabla f, hf)$.

$$(f, \nabla f, hf) \sim \text{GP} \left(\begin{pmatrix} \mu \\ \nabla \mu \\ h\mu \end{pmatrix}, \begin{pmatrix} k & k\nabla^\top & kh^\top \\ \nabla k & \nabla k \nabla^\top & \nabla kh^\top \\ hk & hk \nabla^\top & hkh^\top \end{pmatrix} \right)$$

If $f \sim \text{GP}(\mu \equiv 0, k)$ is sampled from a GP, defined on a compact space X , with $\forall x \neq x' \text{ Var}[\phi - \phi'] = k(x, x') - 2k(x, x') + k(x', x') \neq 0$, then f a.s. has a unique maximum on X .

Adding constraints to a GP, e.g., “ f has a local maximum at x_0 ”, or “ $f'(x_0) = 0, f''(x_0) \leq 0$ ”, can make the process non-Gaussian (with a GP, the distribution of $f''(x_0)$ is unbounded). In this case, use factor graphs.

3. The prior on the mean only affects the extrapolation regime.



For the prior on the mean, use a constant or a linear combination of basis functions, $\mu(x) = \beta' \psi(x)$. *Stationary* covariance functions are of the form $k(x, x') = \kappa(x - x')$; *isotropic* ones of the form $k(x, x') = \kappa(\|x' - x\|)$.

Stationary covariance functions are characterized by their Fourier transform ν (Bochner's theorem)

$$\kappa(\mathbf{x}) = \int \exp(2\pi i \mathbf{x}' \xi) d\nu(\xi)$$

where ν is a finite, positive, symmetric, Borel measure on \mathbf{R}^n (spectral measure).

Common covariance functions include ($d = \|x' - x\|$):

$$\begin{aligned} k_{M^{1/2}} &= \exp(-d) \\ k_{M^{3/2}} &= (1 + \sqrt{3}d) \exp(-\sqrt{3}d) \\ k_{M^{5/2}} &= (1 + \sqrt{5}d + \frac{5}{3}d^2) \exp(-\sqrt{5}d) \\ k_{SE} &= \exp(-\frac{1}{2}d^2) \end{aligned}$$

Their sample paths are a.s. \mathcal{C}^0 , \mathcal{C}^1 , \mathcal{C}^2 , \mathcal{C}^∞ .

The *spectral mixture* covariance is defined by

$$\begin{aligned} k(\xi) &= \sum_i w_i N(\xi; \mu_i, \Sigma_i) \\ \kappa(\xi) &= \frac{1}{2} [k(\xi) + k(-\xi)] \\ K_{SM}(x, x') &= \sum w_i \cdot \exp[-2\pi^2 (x - x')^\top \Sigma_i (x - x')] \\ &\quad \cdot \cos[2\pi (x - x')^\top \mu_i]. \end{aligned}$$

The *linear* covariance function corresponds to

$$\begin{aligned} f(x) &= \beta_0 + \beta^\top \mathbf{x} \\ \beta &\sim N(a, B) \\ k_{LIN}(x, x') &= b^2 + x^\top B x \end{aligned}$$

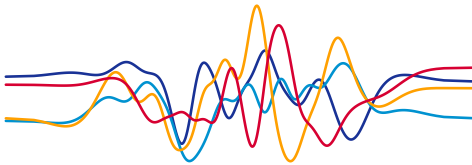
We can rescale the input and the output of the covariance function: $\lambda^2 k(d/\ell)$.

Automatic relevance determination (ARD) estimates those length and scale parameters from data.

Beta warping transforms the data with

$$\begin{cases} \mathbf{R} \longrightarrow [0, 1] \\ x \longmapsto \text{Beta}(x; a, b) \end{cases}$$

to focus around zero.



Covariance functions can be combined:

$$\begin{aligned} \text{Cov}[f + g] &= k_f + k_g \\ \text{Cov}[fg] &= k_f k_g. \end{aligned}$$

4. To select the GP family, the input and output scales, and the observation noise, use the MAP estimator, or model averaging (hyperparameter marginalization, perhaps via MCMC), or search through a space of covariance structures (e.g., described with a grammar), perhaps with Bayesian optimization.

5. (One-step) *Bayesian decision theory* suggests to select the action maximizing the expected utility

$$\begin{aligned} \mathcal{D} &: \text{observed action} \\ \psi &: \text{unobserved data} \\ a \in \mathcal{A} &: \text{actions} \\ u(a, \psi, \mathcal{D}) &: \text{utility} \\ \text{Maximize}_a \mathbb{E}[u(a, \psi, \mathcal{D}) \mid a, \mathcal{D}] \end{aligned}$$

For $T = 1$, we maximize

$$\alpha_1(x, \mathcal{D}) = \mathbb{E}[u(\mathcal{D}_1) \mid x] - u(\mathcal{D}).$$

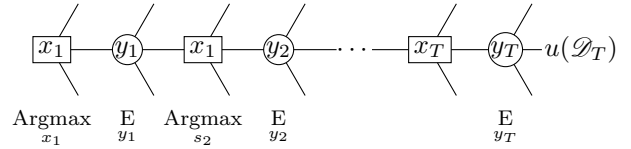
For $T = 2$, we would like to maximize

$$\begin{aligned} \alpha_2(x, \mathcal{D}) &= \mathbb{E}[u(\mathcal{D}_2) - u(\mathcal{D}) \mid x_1] \\ &= \mathbb{E}[u(\mathcal{D}_2) - u(\mathcal{D}_1) + u(\mathcal{D}_1) - u(\mathcal{D}) \mid x_1] \\ &= \mathbb{E}[\alpha_1(x_2 \mid \mathcal{D}_1) \mid x_1 = x] + \alpha_1(x, \mathcal{D}) \end{aligned}$$

but we know that, at the next step, we will choose x_2 to maximize α , so

$$\alpha_2(x, \mathcal{D}) = \mathbb{E}[\alpha_1^*(\mathcal{D}_1) \mid x_1 = x] + \alpha_1(x, \mathcal{D}).$$

There is an optimization, α_1^* , parametrized by x_1 , inside an expectation – and, if we increase T , there will be even more nesting.



$$\begin{aligned} \alpha_T &= \alpha_1 + \mathbb{E}[\alpha_{T-1}^*] \\ &= \alpha_1 + \mathbb{E}[\text{Max } \alpha_{T-1}] \\ &= \alpha_1 + \mathbb{E}[\text{Max } \mathbb{E}[\text{Max } \alpha_{T-2}]] \\ &\dots \end{aligned}$$

Instead, we can use an optimal but truncated path ($T = 1$ or 2), or a suboptimal but complete path (using a heuristic to select x_t).

6. For Bayesian optimization, the final action is a bet on the location of the extremum. The action space can be:

- Only the points examined;
- Points close to the points examined;
- All the points in the domain \mathcal{X} of f .

For linear (risk-neutral) utility:

$$\begin{aligned} u(\mathcal{D}) &= \text{Max}_{(x,y) \in \mathcal{D}} \mu_{\mathcal{D}}(x) && \text{single reward} \\ u(\mathcal{D}) &= \text{Max}_{(x,y) \in \mathcal{X}} \mu_{\mathcal{D}}(x) && \text{global reward} \end{aligned}$$

(do not use $\max_{(x,y) \in \mathcal{D}} y$: it includes observation noise).

For risk-averse or risk-seeking behaviour, replace μ with $\mu + \beta\sigma$, $\beta < 0$ or $\beta > 0$.

Optimal control is slightly different: we want each y_i to be large, *i.e.*, we are maximizing $\sum y_i$.

The information gain, $E[\Delta H]$, or $E[\Delta \text{KL}]$, is another utility function.

7. The expected gain utility

$$\alpha(x, \mathcal{D}) = E[u(\mathcal{D}')|x, \mathcal{D}] - u(\mathcal{D})$$

can be used as an activation function.

Utility	Activation
single reward	expected improvement
global reward	knowledge gradient
unit for improving single reward	probability of improvement
information gain	mutual information
cummulated reward	posterior mean

Bayesian optimization is based on either Bayesian decision theory, or multi-arm bandits (infinite-arm bandits), for instance, using an upper bound (e.g., the 99.9% quantile) of $y = f(x)$; it is equivalent to maximizing the probability of improvement beyond some threshold.

Thompson sampling maximizes a random acquisition function, sampled from the posterior.

Two-step lookahead works better.

8. For Gaussian processes, without noise, the acquisition functions without noise are

$$\alpha_{\text{EI}}(x) = (\mu - \phi^*)\Phi\left(\frac{\mu - \phi^*}{\sigma}\right) + \sigma\phi\left(\frac{\mu - \phi^*}{\sigma}\right)$$

$$\alpha_{\text{PI}} = \Phi\left(\frac{\mu - \tau}{\sigma}\right)$$

where

$$\begin{aligned} \mu &= \mu(x) \\ \sigma &= \sigma(x) \\ \phi^* &: \text{best value of } f(x) \text{ so far} \\ \Phi, \phi &: \text{Gaussian cdf and pdf} \\ \tau &: \text{improvement threshold} \end{aligned}$$

9. For the implementation, use the Coolesky decomposition, and update it sequentially.

Given observations z , use *inducing values* v

$$\begin{pmatrix} v \\ z \end{pmatrix} \sim N\left(\begin{pmatrix} \mu \\ m \end{pmatrix}, \begin{pmatrix} \Sigma & k' \\ k & C + N \end{pmatrix}\right)$$

and compute an approximation of their conditional distribution

$$v|z \sim N(\tilde{\mu}, \tilde{\Sigma})$$

and marginalize them

$$p(f|z) = \int p(f|z)q(v|z)dz.$$

11. We assumed that each observation had a constant cost; if it varies, and is known in advance, estimate it (with a GP, jointly).

For optimal batch selection, try an n -dimensional acquisition function.

Multifidelity optimization also uses several surrogates of an expensive objective function (e.g., early stopping – freeze-thaw, hyperband).

For multi-objective optimization, use the volume under an estimate of the Pareto frontier as utility (S -metric) and maximize the expected hypervolume improvement (EHVI).

On the origin of implicit regularization in stochastic gradient descent S.L. Smith et al. (2021)

Full-batch gradient descent with *finite* step size can be described as a modified gradient flow, with a regularizer $\frac{1}{4}\alpha \|\nabla \ell\|^2$, where α is the learning rate. For SGD, the implicit regularization is

$$\frac{\alpha}{4m} \sum \|\nabla \ell_i\|^2.$$

Implicit gradient regularization D.G.T. Barrett and B. Dherin (2021)

The *discrete* steps of gradient descent lead the model away from the steepest descent path on the loss landscape. This path can be described as the steepest descent path on a modified loss landscape. The modified loss uses a gradient penalty $\frac{\alpha}{4} \sum \|\nabla \ell_i\|^2$.

The implicit regularization of stochastic gradient flow for least squares A. Ali et al. (2020)

Gradient descent for least squares

$$\text{Minimize}_{\beta} \frac{1}{2n} \|y - X\beta\|^2$$

with infinitesimal step sizes is described by the *gradient flow*

$$\dot{\beta}_t = \frac{1}{n} X'(y - X\beta_t), \quad \beta_0 = 0.$$

Stochastic gradient descent, with minibatches I of size $|I| = m$, can be described by a SDE

$$d\beta = \frac{1}{n} X'(y - X\beta)dt + \varepsilon Q(\beta)^{1/2} dW$$

$$Q(\beta) = \text{Cov}_I \left[\frac{1}{m} X'_I(y_I - X_I\beta) \right].$$

It is similar to ridge regression with $\lambda = 1/t$.

The benefits of implicit regularization from SGD in least squares problems
Z. Zou et al. (2021)

(Unregularized) *stochastic* gradient descent has a regularizing effect: it generalizes at least as well as ridge regression.

Implicit regularization of discrete gradient dynamics in linear neural networks
G. Gidel et al. (2019)

The optimization of an overparametrized model does not end in any parameter achieving a zero training loss: the choice of optimization algorithm (and of hyperparameters) directs training towards a particular solution; for a deep *linear* model, gradient descent leads to a low-rank solution.

On lazy training in differentiable programming
L. Chizat et al (2019)

Neural nets sometimes converge to zero training loss with their parameters hardly varying: this is due to the scaling of the weights (or, equivalently, of the input), which make the network behave as its linearization around its initialization.

Robust supervised learning with coordinate gradient descent
S. Gaïffas and I. Merad (2022)

Estimating a robust gradient $\nabla_{\theta} \ell$ is time-consuming: instead, use robust estimates of the partial derivatives $E[\partial \ell / \partial \theta_i]$, with coordinate descent. Choose which coordinate to update with importance sampling.

Robust estimators of the mean of x_1, \dots, x_n include the *median-of-means* (MOM) (median of the blockwise means), the *trimmed mean*, and the *Catoni-Holland estimator*: μ such that

$$\sum \psi \left(\frac{x_i - \mu}{\sigma} \right) = 0,$$

where ψ is, e.g., the sigmoid

$$\psi(x) = 2 \operatorname{Arctan}(e^x) - \frac{\pi}{2}$$

and σ satisfies

$$\sum \chi \left(\frac{x_i - \bar{x}}{\sigma} \right) = 0$$

where \bar{x} is the sample mean, and

$$\chi(u) = \frac{u^2}{1 + u^2} - c,$$

where c is such that $E_{Z \sim N(0,1)}[\chi(Z)] = 0$. Both μ and σ can be computed with fixed point iterations:

$$\begin{aligned} \mu &\leftarrow \mu + \frac{\sigma}{\mu} \sum \psi \left(\frac{x_i - \mu}{\sigma} \right) \\ \sigma &\leftarrow \sigma \left(1 - \frac{\chi(0)}{n} \sum \chi \left(\frac{x_i - \bar{x}}{\sigma} \right) \right). \end{aligned}$$

Alternatively, one could use traditional empirical risk minimization (ERM) on disjoint subsets, and aggregate them with a median-of-mean.

A generalized Catoni's M-estimator under finite α th moment assumption with $\alpha \in (1, 2)$
P. Chen et al.

Try M-estimators of location

$$\sum_i \phi(\beta(x_i - \hat{\theta})) = 0$$

with

$$\phi(x) = \begin{cases} \log\left(1 + x + \frac{x^\alpha}{\alpha}\right) & \text{if } x \geq 0 \\ -\log\left(1 - x + \frac{x^\alpha}{\alpha}\right) & \text{if } x < 0 \end{cases}$$

Efficient learning with robust gradient descent
M.J. Holland and K. Ikeda

In gradient descent, replace the average of the gradients with an M-estimator.

Challenging the empirical mean and empirical variance: a deviation study
O. Catoni (2011)

Properties of the M-estimator of the mean

$$\sum \psi(\alpha(y_i - \hat{\theta})) = 0$$

with ψ non-decreasing and

$$-\log\left(1 - x + \frac{x^2}{2}\right) \leq \psi(x) \leq \log\left(1 + x + \frac{x^2}{2}\right).$$

Mean estimation and regression under heavy-tailed distributions – a survey
G. Lugosi and S. Mendelson (2019)

The quality of a (univariate) mean estimator $\hat{\mu}_n$ is often measured with the mean square error $E[(\hat{\mu}_n - \mu)^2]$ but, if the differences $|\hat{\mu}_n - \mu|$ are not sufficiently concentrated, the expectation need not reflect the typical error. Instead, prefer estimators $\hat{\mu}_n$ that are close to μ with high probability

$$|\hat{\mu}_n - \mu| \leq \varepsilon \text{ w.p. } 1 - \delta.$$

For Gaussian data, the sample mean satisfies (L -sub-Gaussian)

$$|\hat{\mu}_n - \mu| \leq \frac{L\sigma\sqrt{\log(2/\delta)}}{\sqrt{n}} \text{ w.p. } 1 - \delta.$$

We want an estimator with a similar inequality, but for fat tail distributions (the estimators will depend on δ , but will only require finite variance; with more assumptions, e.g., finite third moment, they need no longer depend on δ):

- The *median-of-means* estimator splits the data into k groups, computes their means, and the median of those means;
- *Catoni's* estimator generalizes the characterization of the mean as the solution $\hat{\mu}_n$ of $\sum (x_i - \hat{\mu}_n) = 0$ to

$$\sum \psi \left(\frac{x_i - \hat{\mu}_n}{s} \right) = 0$$

for

$$\phi(x) = \begin{cases} \log \left(1 + x + \frac{x^\alpha}{\alpha} \right) & \text{if } x \geq 0 \\ -\log \left(1 - x + \frac{x^\alpha}{\alpha} \right) & \text{if } x < 0 \end{cases},$$

$s = \sqrt{n/2}\sigma$, and some estimator of σ ;

- The *trimmed mean* splits the data in two, uses the first half to compute the quantiles, and the second half (without the observations beyond the quantiles) to compute the mean.

For multivariate sub-Gaussian data,

$$\|\hat{\mu}_n - \mu\| \leq \sqrt{\frac{\text{tr } \Sigma}{n}} + \sqrt{\frac{2\lambda_{\max} \log(1/\delta)}{n}} \text{ w.p. } 1 - \delta$$

$$\mathbb{E} \|\hat{\mu}_n - \mu\|^2 = \frac{\text{tr } \Sigma}{n}$$

Traditional multivariate medians (geometric (spatial), Tukey (half-space), Liu (simplicial), Oja) do not give the desired bounds. Neither do the following medians:

- The point $\hat{\mu}_n$ such that the Euclidean ball centered at $\hat{\mu}_n$ that contains half the data has minimal radius;
- Rescale the x_i 's such that $\|x_i\| \leq 1/\alpha$ (i.e., shrink them towards zero), perhaps after centering the data (use independent batches to estimate the center (mean) and to compute the shrunk mean).

The following (*median of means tournament*) works: let T_a , $a \in \mathbf{R}^n$, be the set of points $x \in \mathbf{R}^n$ at least as close to z_1, \dots, z_k as a , and set $\hat{\mu}_n = \text{Argmin}_a \text{radius}(T_a)$, where z_1, \dots, z_k are block means (a semi-definite relaxation can be computed efficiently).

Robust sparse voting **Y. Allouah et al. (2022)**

The *quadratically regularized weighted median* is

$$\text{QrMed}_W(w, x) = \text{Argmin}_z \frac{1}{2} W z^2 + \sum_i w_i |z - x_i|.$$

Contrary to the median, it is W -Byzantine resilient (an attacker can change the median by choosing to be on one end or the other, and its impact is unbounded).

The *Byzantine resilient mean estimator* is a clipped

mean centered on the QrMed:

$$\text{BrMean}_W(w, x) =$$

$$\text{ClipMean} \left(w, x; \text{QrMed}_{4W}(w, x), \frac{\|w\|_1}{4W} \right)$$

$$\text{ClipMean}(w, x; \mu, \Delta) =$$

$$\frac{1}{\|w\|_1} \sum_i w_i \text{Clip}(x_i, \mu - \Delta, \mu + \Delta).$$

To aggregate complete rankings, use the majority judgement, or the randomized Condorcet voting system.

If the voters only provide sparse scores, on arbitrary scales, the following is sparsely unanimous and W -Byzantine resilient (but not independent of irrelevant alternatives).

n : voter

a : alternative

θ_{na} : scores

w_n : voting rights

$$\theta_a \leftarrow \frac{\theta_a - \text{Min } \theta_{am}}{\text{Max } \theta_{am} - \text{Min } \theta_{am}}$$

$$s_{nm} = \text{Mean}_{a,b} \frac{\theta_{ma} - \theta_{mb}}{\theta_{na} - \theta_{nb}}$$

$$s_n = 1 + \text{BrMean}_m(s_{nm} - 1)$$

$$\tau_{nm} = \text{Mean}_a s_m \theta_{ma} - s_n \theta_{na}$$

$$\tau_m = \text{BrMean}_m \tau_{nm}$$

$$\text{Mehestan}(w, \theta) = \text{QrMed}_n s_n \theta_n + \tau_n$$

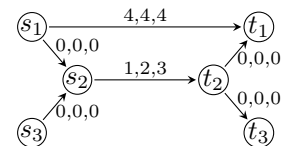
(if you do not insist on resilience and scale invariance, the weighted mean is better).

Individually scaling the preferences cannot lead to sparse unanimity.

Algorithmic game theory

T. Kesselheim (2020)

1. In a *network congestion game*, n players have to find a path from their respective start nodes to their target node, on a graph, and incur a cost (delay) for each edge used; the per-user cost depends on the number of users using this edge.



Each player wants to minimize its cost, but society wants to minimize the total cost.

A *congestion game* is the datum of

- A set of players $\llbracket 1, n \rrbracket$;
- A set of resources $R = \llbracket 1, m \rrbracket$;
- A strategy space for each player $\Sigma_i \subset 2^R$;

- A delay function for each resource $d_r : \llbracket 1, n \rrbracket \rightarrow \mathbf{Z}$.

The cost of a state $s = (s_1, \dots, s_n) \in \Sigma_1 \times \dots \times \Sigma_n$ for player i is $c_i(s) = \sum_{r \in S_i} d_r(n_r(s))$, where $n_r(s) = \#\{i : r \in S_i\}$ is the number of players using resource r .

In a congestion game, every sequence of best response improvement steps is finite: there is a pure Nash equilibrium. This can be seen by showing that the *Rosenthal potential*

$$\Phi(s) = \sum_r \sum_{k=1}^{n_r(s)} d_r(k)$$

decreases.

2. This sequence can be exponentially long but, in a *singleton* congestion game ($\forall i \forall R \in S_i \ |R| = 1$), it is at most polynomial.

Pure Nash equilibria are exactly the local minima of the Rosenthal potential.

In symmetric network congestion games (all players have the same source and target nodes), the Nash equilibria can be computed in polynomial time (by reduction to a min-cost flow problem).

3. A (cost-minimization) *normal form game* is defined by

- A set of players $\llbracket 1, n \rrbracket$;
- For each player i , a set of pure strategies S_i ;
- For each player, a cost function

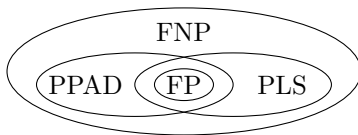
$$c_i : S = \prod_j S_j \rightarrow \mathbf{R}.$$

Bimatrix games are a special case.

	A	B					
A	2	1		L	0		
B	5	1		I	1	6	

4. Pure Nash equilibria do not always exist, but every finite normal form game has a *mixed Nash equilibrium* (a distribution on strategies, for each player); the proof uses Brouwer's fixed point theorem. For bimatrix games, one can find a mixed Nash equilibrium with a simplex-like algorithm.

5. NP is a complexity class for decision problems, but Nash equilibrium is a search problem. A search problem is FP if we can find a solution in polynomial time; it is FNP if we can verify a solution in polynomial time.



Finding a mixed Nash equilibrium in a normal form game is PPAD complete; finding a pure Nash equilibrium in a congestion game is PLS complete.

A local search problem is in PLS (polynomial local search) if there are polynomial algorithms to

- Find a feasible solution;
- Compute the cost of a feasible solution;
- Improve a feasible solution or, if it is already optimal, prove it.

Max-Cut is PLS-complete, and can be reduced to the problem of finding a Nash equilibrium in a congestion game.

6. A *correlated equilibrium* is a probability distribution on states (the players are no longer independent) such that $\forall i \in N \ \forall s_i \in S_i \ \forall s'_i \in S_i$

$$\mathbb{E}_{s \sim p} [c_i(s) | s_i] \leq \mathbb{E}_{s \sim p} [c_i(s'_i, s_{-i}) | s_i],$$

i.e., whenever p tells player i to play s_i , it is the best thing to do.

A *coarse correlated equilibrium* only demands

$$\mathbb{E}_{s \sim p} [c_i(s)] \leq \mathbb{E}_{s \sim p} [c_i(s'_i, s_{-i})],$$

i.e., player i is better off following p than always choosing s'_i .

A correlated equilibrium can be computed in polynomial time (via linear programming).

To find an (approximate) coarse equilibrium, the players can follow a *no-regret* algorithm. With two players, it would be:

- The player picks a distribution p_t over strategies;
- The adversary picks a cost vector ℓ_t ;
- The player picks an action from p_t , incurs the corresponding loss, and gets to know the cost vector.

The player tries to minimize the (external, expected) regret

$$\mathbb{E} \left[\sum_t \ell_t(a_t) - \min_a \sum_t \ell_t(a) \right]$$

(we compare the player's action with the best constant action in hindsight – the best action sequence in hindsight, $\sum_t \min_a \ell_t(a)$, would be too strong). For multiple players, each player chooses a distribution, and the loss vector is determined by the decisions of the other players

7. The *multiplicative weights algorithm* progressively updates the probabilities by increasing those with low cost and decreasing those with high cost:

$$w_1 = (1, \dots, 1)$$

$$p_t(i) \propto w_t(i)$$

$$w_{t+1}(i) \leftarrow w_t(i)(1 - \eta)^{\ell_t(i)}.$$

With $\eta = \sqrt{\log M / T}$, the regret is at most $2\sqrt{T \log M}$.

8. The *social cost* is the sum of the players' costs

$$\text{SC}(S) = \sum_i c_i(S).$$

The **price of anarchy** for pure Nash equilibria (PNE) compares the worst Nash equilibrium with the best societal outcome.

$$\text{PoA}_{\text{PNE}} = \frac{\max_{S \in \text{PNE}} \text{SC}(S)}{\min_S \text{SC}(S)}$$

In a congestion game with affine delay functions, $\text{PoA} \leq 5/2$. Other equilibrium concepts have a higher price of anarchy.

$$1 \leq \text{PoA}_{\text{PNE}} \leq \text{PoA}_{\text{MNE}} \leq \text{PoA}_{\text{CE}} \leq \text{PoA}_{\text{CCE}}$$

A game is (λ, μ) -smooth ($\lambda > 0, \mu < 1$) if

$$\forall s, s' \quad \sum_i c_i(s_i^*, s_{-i}) \leq \lambda \text{SC}(s^*) + \mu \text{SC}(s);$$

the price of anarchy for coarse correlated equilibria is then at most $\lambda/(1 - \mu)$.

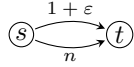
9. The *price of stability* compares the best equilibrium (the PoA used the worst) with the best societal outcome.

$$\text{PoSEq} = \frac{\min_{s \in \text{Eq}} \text{SC}(s)}{\min_s \text{SC}(s)}$$

A *cost-sharing game* is a congestion game with $d_r(x) = c_r/x$; the societal cost is then $\text{SC}(s) = \sum_{r: n_r(s) \geq 1} c_r$. The price of stability of a cost-sharing game is at most

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}.$$

For a symmetric cost sharing game (all players have the same strategies), it is 1, but the price of anarchy can be large.



A *market sharing game* is a (utility maximization) congestion game with $S_i \subset [1, m]$ and $u_i(s) = v_{s_i}/n_{s_i}(s)$, where v_j is the demand in market j , shared among the players entering it. The *social welfare* is then

$$\text{SW}(s) = \sum_{j: n_j(s) \geq 1} v_j.$$

The price of anarchy of a utility-maximization game is

$$\text{PoA}_{\text{Eq}} = \frac{\max_{s \in S} \text{SW}(s)}{\min_{p \in \text{Eq}} \text{SW}(p)}.$$

A utility-maximization game is (λ, μ) -smooth ($\lambda > 0, \mu \geq 0$) if

$$\forall s, s' \quad \sum_i u_i(s_i^*, s_{-i}) \geq \lambda \text{SW}(s^*) - \mu \text{SW}(s);$$

its PoA_{CCE} is at most $(1 + \mu)/\lambda$. The market-sharing game is (1,1)-smooth: $\text{PoA}_{\text{CCE}} \leq 2$.

10. In a *first-price auction*, where each player has a value v_i for the item, makes a bid b_i , and has utility $u_i(b, v_i) = v_i - b_i$ if he wins and zero otherwise, there is a Nash equilibrium: the highest value player bids the second highest value, and the others bid their value.

In a *second price auction* (Vickrey auction), $u_i(b, v_i) = v_i - \text{Max}_{j \neq i} b_j$; it is a dominant strategy to bid truthfully.

Given a set of players $[1, n]$, a set of outcomes X , valuation functions $v_i : X \rightarrow \mathbf{R}$, and possible bids $B = B_1 \times \dots \times B_n$, a *mechanism* is the datum of an outcome rule $f : B \rightarrow X$ and a payment rule $p : B \rightarrow \mathbf{R}^n$. The players' utilities are

$$u_i(b) = v_i(f(b)) - p_i(b).$$

If the bids are valuation functions, the mechanism is *direct*. A mechanism is **truthful** (dominant strategy incentive compatible, DSIC) if bidding the true value is a weakly dominant strategy.

11. In a *single-parameter mechanism*, $X \subset \mathbf{R}_{\geq 0}^n$ and $v_i(x) = v_i \cdot x_i$. In this case, if the allocation rule f is monotone, i.e., the $z \mapsto f_i(z, b_{-i})$ are non-decreasing, setting the payment rule to

$$p_i(b_i, b_{-i}) = b_i f_i(b_i, b_{-i}) - \int_0^{b_i} f_i(t, b_{-i}) dt$$

makes the mechanism truthful (**Myerson's lemma**).

12. In a *combinatorial auction*, there are n items and each bidder has a value for each subset of items. If bidders are *single-minded* (they only care whether they get all their desired items or not), if their preferences are public (but their valuations private), the *greedy-by-value* mechanism (give bidders what they want, if possible, starting bidders with the largest bids) is a d -approximation, where d is the size of the largest bundle. For large bundle sizes, prefer *greedy-my-sqrt-value-density* (order the bidders by $b_i/\sqrt{|S_i^*|}$): it is an \sqrt{m} -approximation.

13. A **VCG mechanism** is a mechanism whose allocation rule $f : V \rightarrow X$ maximizes social welfare, i.e., $f(b) \in \text{Argmax}_x \sum_i b_i(x)$, and whose payment rule is

$$p_i(b) = \text{Max}_x \sum_{j \neq i} b_j(x) - \sum_{j \in S} b_j(f(b))$$

(player i 's externality). It is DSIC. Examples include second-price single-item auction, sponsored search auctions ($v_i(x) = v_i \alpha_j$ if agent i gets slot j), unit-demand combinatorial auctions ($v_i(S) = \text{Max}_{j \in S} v_{ij}$).

VCG requires solving a welfare maximization problem: an approximation is not enough – for instance, VCG payments with a greedy allocation for unit-demand combinatorial auctions (a 2-approximation) do not give a truthful mechanism.

14. Given a (non-truthful) mechanism, with incomplete information (the bidders' values are drawn from a known distribution $v_i \sim \mathcal{D}_i$), a **Bayes-Nash equilibrium** is a profile of bidding functions $\beta_i : V_i \rightarrow B_i$ such that $\forall i \quad \forall v_i \in V_i \quad \forall b'_i \in B_i$

$$\mathbb{E}_{v_{-i} \sim \mathcal{D}_{-i}} u_i(\beta(v), v_i) \geq \mathbb{E}_{v_{-i} \sim \mathcal{D}_{-i}} u_i((b'_i, \beta_{-i}(v)), v_i).$$

15. A mechanism is (λ, μ) -smooth if

$$\forall v \in V \quad \forall i \quad \forall b_i^* \quad \forall b \in B$$

$$\sum_i u_i((b_i^*, b_{-i}), v_i) \geq \lambda \cdot \text{OPT}(v) - \mu \sum_i p_i(b);$$

the price of anarchy is then bounded (same bound for pure Nash equilibria, (coarse) correlated equilibria, and Bayes-Nash equilibria):

$$\text{PoA} \leq \frac{\text{Max}(\mu, 1)}{\lambda}.$$

Single-item first-price auction is $(\frac{1}{2}, 1)$ -smooth; single-item all-pay auction is $(\frac{1}{2}, 2)$ -smooth.

16. Combinatorial auction with unit-demand valuations and first-price payments is $(\frac{1}{2}, 1)$ -smooth. First-price greedy-by-sqrt-value-density for multi-minded combinatorial auctions is $(\frac{1}{2}, \sqrt{2m})$ -smooth.

17. A **Walrasian equilibrium** (in a combinatorial auction) is a *price vector* (Nash equilibria were about players' bids, not prices) $q \in \mathbf{R}_{\geq 0}^m$ and an allocation S such that each bidder gets a bundle maximizing his utility

$$\forall S'_i \quad v_i(S_i) - \sum_{j \in S_i} q_j \geq v_i(S'_i) - \sum_{j \in S'_i} q_j.$$

Walrasian equilibria (if they exist) maximize social welfare. Unit-demand VCG prices are a Walrasian equilibrium.

18. In the *posted prices* mechanism, buyers come one by one, and buy the set of unsold items maximizing their utility. For unit-demand, the expected social welfare of the posted prices mechanism is half the expected optimal welfare – set the prices to (the expectation of) half the value for the buyer getting it in the optimal welfare allocation.

19. If the values are drawn from distributions \mathcal{D}_i , the *virtual values* are

$$\phi_i(t) = t - \frac{1 - F_i(t)}{f_i(t)}.$$

For a truthful single-parameter mechanism, the expected revenue is the expected virtual welfare:

$$\mathbb{E} \sum_i p_i(v) = \mathbb{E} \sum_i \phi_i(v_i) x_i(v).$$

The *virtual welfare maximizing mechanism* (with Myerson's payments) is truthful and maximizes expected revenue among truthful payments.

20. The second-price auction maximizes the expected revenue among all truthful mechanisms (if valuations are drawn from the same distribution, if the item is always allocated). Posting a price

$$p^* = \text{Max}\{\phi^{-1}(0), F^{-1}(1 - \frac{1}{n})\}$$

gives a $1 - (1 - \frac{1}{n})^n \geq (1 - 1/e)$ -approximation of the optimal expected revenue. (Surprisingly, truthfulness is not a restriction.)

21. To match agents to houses, when agents have different preferences (without money), start by assigning the houses at random, then build a directed graph with

edges from each agent to the owner of the house he prefers best among the remaining ones, find a cycle, reallocate the houses in it, and remove the corresponding nodes; iterate until the graph is empty. The result allocation is *stable*.

22. The **Gale-Shapley** algorithm generalizes this to situations with preferences from both agents and items.

23. To cut a cake $[0, 1]$ between n agents with valuations

$$V_i(X) = \int_{x \in X} v_i(x) dx \quad X \subset [0, 1] \quad V_i([0, 1]) = 1,$$

we may want:

- Proportionality: $\forall i \quad V_i(A_i) \geq 1/n$;
- Envy-freeness: $\forall i, j \quad V_i(A_i) \geq V_i(A_j)$;
- Equitability: $\forall i, j \quad V_i(A_i) = V_i(A_j)$.

For two agents, the cut-and-choose mechanism is proportional, envy-free, but not necessarily equitable. For n agents, the *Dubins-Spanier algorithm* is proportional. Ensuring envy-freeness is more complicated. None of those algorithms is truthful.

24. A *ranking rule* for three or more candidates satisfying unanimity and independence of irrelevant alternatives is a dictatorship.

25. **Shapley values** formalize cost-sharing.

Mechanics for mathematicians J. Wunsch (2020)

2. Newton's law, $F = ma$, describes the evolution of a physical system as a second-order differential equation. Examples include:

- Gravitation (1d): $\ddot{x} = -1$;
- Spring: $\ddot{x} = -\omega^2 x$;
- Magnetism: $\ddot{x} = \dot{x} \times B$;
- Gravity: $\ddot{x} = -\hat{x} / \|x\|^2$.

3. The initial value problem has a unique solution (Picard-Lindelöf); it may not have a closed form solution: try Taylor expansions or numeric methods (Newton).

4. In dimension 1, if the force does not depend on time, it comes from a **potential**:

$$\ddot{x} = F(x) \rightsquigarrow \ddot{x}x = F(x)\dot{x} \rightsquigarrow \frac{d}{dt} \left(\frac{1}{2} \dot{x}^2 + V(x) \right) = 0$$

where $V'(x) = -F(x)$. The energy $E = \frac{1}{2} \dot{x}^2 + V(x)$ is constant.

5. Plot the level curves of the energy in the phase plane to identify periodic motion and distinguish between stable and unstable equilibria ($V(x_0) > 0$ and $V(x_0) < 0$).

6. In \mathbf{R}^n , we would also like to write $F(x) = -\nabla V(x)$, but that is only possible if $\text{curl } F = 0$, or, more generally, if the vector field F is conservative, *i.e.*, $\int_{\gamma} F(x) \cdot dx$ only depends on the endpoints of the path γ .

7a. The **Lagrangian** is the difference between kinetic T and potential V energy.

$$L = T - V \quad L(x, \dot{x}) = \frac{1}{2} |\dot{x}|^2 - V(x)$$

The *action* of a trajectory $\gamma : [0, T] \rightarrow \mathbf{R}^3$ is

$$S(\gamma) = \int_0^T L(\gamma(t), \dot{\gamma}(t)) dt.$$

The *principle of least action* states that the action is stationary, *i.e.*, its directional derivatives are zero:

$$\left. \frac{d}{ds} S(\gamma + s\phi) \right|_{s=0} = 0$$

for all paths ϕ with $\phi(0) = \phi(T) = 0$.

For a functional of the form

$$S(\gamma) = \int_0^T L(\gamma(t), \dot{\gamma}(t), t) dt,$$

the stationarity condition becomes

$$\int_0^T \left[\frac{\partial L}{\partial x} \cdot \phi(t) + \frac{\partial L}{\partial v} \cdot \phi'(t) \right] dt = 0$$

and, after integrating the second term by parts (and using $\phi(0) = \phi(T) = 0$),

$$\int_0^T \left(\frac{\partial L}{\partial x} - \frac{d}{dt} \frac{\partial L}{\partial v} \right) \cdot \phi(t) dt = 0$$

giving the **Euler-Lagrange equation**

$$\frac{\partial L}{\partial x} = \frac{d}{dt} \frac{\partial L}{\partial v}.$$

7b. If a coordinate x_i does not appear in L , then

$$\frac{d}{dt} \frac{\partial L}{\partial v_i} = \frac{\partial L}{\partial x_i} = 0,$$

i.e., $\partial L / \partial v_i$ is a conserved quantity.

There is a similar result for time: if $\partial L / \partial t = 0$,

$$\begin{aligned} \frac{dL}{dt} &= \frac{\partial L}{\partial x} \cdot \dot{\gamma} + \frac{\partial L}{\partial v} \cdot \ddot{\gamma} + \cancel{\frac{\partial L}{\partial t}} \\ &= \left(\frac{d}{dt} \frac{\partial L}{\partial v} \right) \cdot \dot{\gamma} + \frac{\partial L}{\partial v} \cdot \ddot{\gamma} \quad (\text{Euler-Lagrange}) \\ &= \frac{d}{dt} \left(\frac{\partial L}{\partial v} \cdot \dot{\gamma} \right) \quad (\text{product rule}) \end{aligned}$$

hence

$$\frac{d}{dt} \left(-L + \frac{\partial L}{\partial v} \cdot \dot{\gamma} \right) = 0$$

i.e., the *Hamiltonian*

$$H(x, v) = -L(x, v) + \frac{\partial L}{\partial v} \cdot v$$

is conserved. (In simple cases, $H = T + V$.)

The Lagrangian approach facilitates changes of coordinates and easily allows for constraints (e.g., with a reparametrization).

7c. More generally, a continuous symmetry

$$\begin{aligned} F^a : \mathbf{R}^n &\longrightarrow \mathbf{R}^n \\ F_*^a(x, v) &= (F(x), \nabla F(x) \cdot v) \\ L(F_*^a(x, v), t) &= L(x, v, t) \end{aligned}$$

gives rise to a conservation law (**Noether's theorem**)

$$p = \frac{\partial L}{\partial v} \cdot W \quad \text{where } W(x) = \left. \frac{d}{da} F^a(x) \right|_{a=0}.$$

Examples include momentum and angular momentum.

11. The Hamiltonian is defined as

$$H = \frac{\partial L}{\partial v} \cdot v - L.$$

With the change of variables

$$\begin{aligned} q &= x \\ p &= \frac{\partial L}{\partial v} \end{aligned}$$

it becomes $H = pv - L(x, v)$. We then have

$$\begin{aligned} \frac{\partial H}{\partial p} &= v + p \cancel{\frac{\partial}{\partial p}} - \cancel{\frac{\partial L}{\partial x} \frac{\partial x}{\partial p}} - \cancel{\frac{\partial L}{\partial v} \frac{\partial v}{\partial p}} \\ &= v \\ &= \dot{x} \\ &= \dot{q} \\ \frac{\partial H}{\partial q} &= p \cancel{\frac{\partial}{\partial p}} - \frac{\partial L}{\partial x} \frac{\partial x}{\partial p} - \cancel{\frac{\partial L}{\partial v} \frac{\partial v}{\partial p}} \\ &= -\frac{\partial L}{\partial x} \\ &= -\frac{d}{dt} \frac{\partial L}{\partial v} \quad (\text{Euler-Lagrange}) \\ &= -\dot{p} \end{aligned}$$

Hamilton's equations of motion are

$$\begin{aligned} \dot{q} &= \frac{\partial H}{\partial p} \\ \dot{p} &= -\frac{\partial H}{\partial q}. \end{aligned}$$

They define a vector field

$$v_H = \begin{pmatrix} \partial H / \partial p \\ -\partial H / \partial q \end{pmatrix}$$

and a flow $(\Phi_t)_t$.

The Hamiltonian vector field has vanishing divergence, $\nabla \cdot v_H = 0$; its flow is therefore volume-preserving (Liouville's theorem).

For a function defined on phase space, $a(q, p)$,

$$\begin{aligned} \frac{da}{dt} &= \frac{\partial a}{\partial q} \cdot \dot{q} + \frac{\partial a}{\partial p} \cdot \dot{p} \\ &= \frac{\partial a}{\partial q} \cdot \frac{\partial H}{\partial p} - \frac{\partial a}{\partial p} \cdot \frac{\partial H}{\partial q} \\ &= \{a, H\} \quad (\text{Poisson bracket}) \\ &= (v_H \cdot \nabla)a. \end{aligned}$$

The flow is a symplectomorphism,

$$(\nabla \Phi_t)J(\nabla \Phi_t)' = J$$

where $J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}$.

Portfolio construction with Gaussian mixture returns and exponential utility via convex optimization
E. Luxenberg and S. Boyd (2022)

The expected utility of a portfolio whose constituent returns are a Gaussian mixture has a simple closed form

$$\begin{aligned} r &\sim \text{GM}((\mu_i, \Sigma_i, \pi_i)_{1 \leq i \leq k}) \\ R &= w'r \\ U(r) &= 1 - e^{-\gamma R} \\ \mathbb{E}[U(R)] &= 1 - M(w, -\gamma) \\ M(w, t) &= \mathbb{E}[e^{tR}] = \sum \pi_i \exp\left[t\mu_i'w + \frac{1}{2}w'\Sigma_i w\right] \end{aligned}$$

and maximizing the *entropic value at risk*

$$\text{EVaR}_\alpha(R) = \inf_{\lambda > 0} \frac{\log M(w, -\lambda) - \log \alpha}{\lambda}$$

and can also be formulated as a convex problem. [Implementation in a few lines of `cvxpy`.]

Entropic portfolio optimization: a disciplined convex programming framework
D. Cajas (2021)

The EVaR (and the EDaR, entropic drawdown at risk) can also be optimized using samples. Implementation in `Riskfolio-Lib`.

OWA portfolio optimization: a disciplined convex programming framework
D. Cajas (2021)

The *ordered weighted average* (OWA) of $(y_i)_{1 \leq i \leq T}$ is a weighted average of order statistics, $\sum w_i y_{(i)}$. Examples include the *Gini mean deviation*

$$\text{GMD} = \text{Mean}_{i \neq j} |y_i - y_j|$$

or the *L-moments* ($\lambda_2 = \frac{1}{2}\text{GMD}$). They can be formulated with a linear program.

$$\begin{aligned} \text{Find} & \quad w, \alpha, \beta, y \\ \text{To maximize} & \quad w'\mu - \lambda \sum_i (\alpha_i + \beta_i) \\ \text{Such that} & \quad w'\mathbf{1} = 1 \\ & \quad w \neq 0 \\ & \quad w'r = y \\ & \quad \forall i, j \quad \alpha_i + \beta_j \geq w_i y_{(j)} \end{aligned}$$

where r_{ki} is the return of asset k in scenario i .

Causal matrix completion
A. Agarwal et al.

To solve the matrix completion problem

$$\begin{aligned} A &\in \mathbf{R}^{m \times n} && \text{latent} \\ \varepsilon &\in \mathbf{R}^{m \times n} && \text{noise} \\ \mathbb{E}[\varepsilon] &= 0 \\ \tilde{Y} &= A + \varepsilon \\ P &\in [0, 1]^{m \times n} && \text{propensity score} \\ D &\in \{0, 1\}^{m \times n} && \text{missingness mask} \\ \mathbb{E}[D] &= P \\ Y_{ij} &= \begin{cases} \tilde{Y}_{ij} & \text{if } D_{ij} = 1 \\ * & \text{otherwise} \end{cases} \end{aligned}$$

one can try

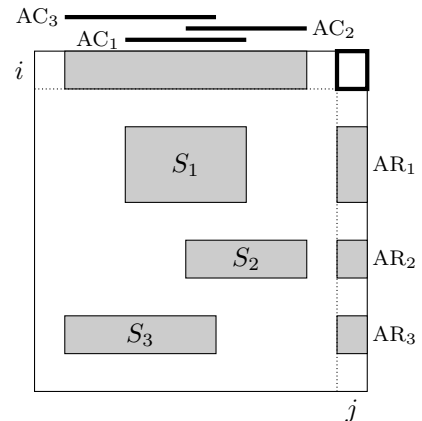
$$\begin{aligned} \text{USVT} & \quad \text{Minimize}_{Q: \text{rank } Q \leq \mu} \| (Y - Q) \odot D \|_2^2 \\ \text{softImpute} & \quad \text{Minimize}_Q \| (Y - Q) \odot D \|_2^2 + \lambda \|Q\|_* \end{aligned}$$

(but they may not perform well if the data is not MCAR) or inverse propensity weighting

$$\text{Minimize}_Q \sum_{i,j: D_{ij}=1} \frac{1}{P_{ij}} d(Y_{ij}, Q_{ij}) + \lambda R(Q).$$

To fill in a missing value Y_{ij} , *synthetic nearest neighbours*

- Look for several fully-observed blocks S_k , with non-overlapping rows (e.g., *maximal bicliques*),
- Approximate the row AC_k from S_k , with a principal component regression,
- Compute the corresponding prediction for Y_{ij} ,
- And average those predictions.



Matrix estimation by universal singular value thresholding
S. Chatterjee (2015)

Given a matrix $X \in [-1, 1]^{m \times n}$, $m \geq n$, with missing values, USVT completes it as follows:

- Fill in the missing values with 0;

- Compute the SVD;
- Only keep the singular values above $2\sqrt{np}$, where p is the proportion of observed values;
- Reconstruct the matrix;
- Clip the entries to $[-1, 1]$.

Applications include:

- Low-rank matrices;
- Latent space models $x_{ij} = f(\beta_i, \beta_j) + \varepsilon_{ij}$;
- Bradley-Terry models

$$p_{ij} = P[\text{team } i \text{ wins over } j] = \frac{a_i}{a_i + a_j}$$

$$X_{ij} \sim \text{Bernoulli}(p_{ij})$$

and its non-linear generalizations (p monotonic);

- Stochastic block models
- Graphons (estimation of a graphon from a single graph sampled from it);
- Distance matrices;
- Positive semi-definite matrices.

**Matrix completion and low-rank SVD
via fast alternating least squares
T. Hastie et al. (2015)**

The matrix completion problem

$$\text{Minimize}_M \frac{1}{2} \|(X - M) \odot O\|_F^2 + \lambda \|M\|_*$$

where $\|\cdot\|_*$ is the nuclear norm, the sum of the singular values (a convex relaxation of the rank) can be solved with the *softImpute* algorithm, iterating:

$$\hat{X} = \text{coalesce}(X, 0)$$

$$\hat{X} \stackrel{\text{SVD}}{=} UDV'$$

$$\hat{M} = US_\lambda(D)V'$$

$$\hat{X} = \text{coalesce}(X, \hat{M})$$

where the soft thresholding is $S_\lambda(D_{ii}) = (D_{ii} - \lambda)_+$; an efficient implementation can use a reduced rank SVD, and write \hat{X} as the sum of a sparse and a low-rank matrix, $\hat{X} = \text{coalesce}(X, \hat{M}) = (X - \hat{M}) + \hat{M}$.

The *maximum margin matrix factorization* (MMMF) problem

$$\text{Minimize}_{A,B} \frac{1}{2} \|(X - AB') \odot O\|_F^2 + \frac{\lambda}{2} (\|A\|_F^2 + \|B\|_F^2)$$

is biconvex and can be solved with alternating least squares (ALS); the solution is

$$\hat{A} = U_r S_\lambda(D_r)^{1/2}$$

$$\hat{B} = V_r S_\lambda(D_r)^{1/2}.$$

The *softImpute-ALS* algorithm combines both approaches.

$$\text{Minimize}_{A,B} \frac{1}{2} \|\hat{X} - AB'\|_F^2 + \frac{\lambda}{2} (\|A\|_F^2 + \|B\|_F^2).$$

A swiss army infinitesimal jackknife

R. Giordano et al. (2020)

To compute approximate bootstrap, cross-validation or jackknife samples for an estimator θ defined by

$$\frac{1}{N} \sum_n w_n g_n(\theta) = 0$$

(the g_n 's could be the gradients of a loss function – use automatic differentiation to compute them) use a first order approximation.

$$G(\theta, w) = \frac{1}{N} w' g$$

$$H(\theta, w) = \frac{1}{N} w' \nabla_\theta g$$

$$G(\theta(w), w) = 0$$

$$\frac{\partial G}{\partial \theta} \frac{\partial \theta}{\partial w} + \frac{\partial G}{\partial w} = 0$$

$$\frac{\partial \theta}{\partial w} = - \left(\frac{\partial G}{\partial \theta} \right)^{-1} \frac{\partial G}{\partial w}$$

$$= -H^{-1} \left(\frac{1}{N} g \right)$$

$$\hat{\theta}_{IJ}(w) = \hat{\theta}_1 + \frac{\partial \theta}{\partial w} \Big|_{w=1} (w - 1)$$

$$= \hat{\theta}_1 - H_1^{-1} G(\hat{\theta}_1, w - 1)$$

**Uncertainty prediction for deep sequential
regression using meta models**

J. Navrátil et al.

A *metamodel* predicts the error of a base model:

$$\text{Base model: } \text{Minimize}_{\phi} E[\text{loss}(\hat{y}, y)]$$

$$\text{Meta model: } \text{Minimize}_{\gamma} E[\text{loss}(\hat{z}, z)]$$

where $z = \text{loss}(\hat{y}, y)$ (those three loss functions can be different).

**Learning prediction intervals
for model performance**

B. Elder et al. (2021)

Use a meta model to predict the performance (e.g., accuracy) of the base model, and a meta-meta model to predict a confidence interval on it.

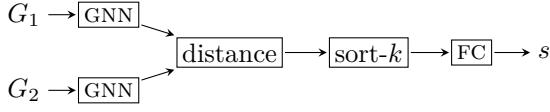
**Confidence scoring using whitebox
meta-models with linear classifiers**

T. Chen et al.

Build a meta model from linear features on the various layers of the base model.

Graph similarity learning for change point detection in dynamic networks
D. Sulem et al. (2022)

To detect changepoints in time-varying networks $(G_t)_{t \geq 0}$, train a Siamese GNN



and use its output as a similarity measure, e.g., $\text{Mean}_{i \in [1, L]} s(G_t, G_{t-i})$ or $\text{Mean}_{\substack{i, j \in [1, L] \\ i < j}} s(G_{t-i}, G_{t-j})$.

Sorted pooling in convolutional networks for one-shot learning
H. András (2020)

Max pooling can be replaced by:

- k -Max pooling, returning the k -th largest value in each patch;
- Sort- k pooling, returning a weighted average of the top- k values, with learned weights.

Rough volatility: fact or artefact
R. Cont and P. Das (2022)

Volatility clustering suggests $H > \frac{1}{2}$, even though the roughness of the realized volatility suggests $H < \frac{1}{2}$.

The p -variation of a function $x : [0, T] \rightarrow \mathbf{R}$ is the limit

$$\sum |x(t_{i+1}) - x(t_i)|^p \longrightarrow [x]^p$$

when the partition $(0, t_0, t_1, \dots, t_N = T)$ gets finer. The *variation index* and the *roughness index* are then

$$p(x) = \inf \{ p \geq 1 : [x]^p < \infty \}$$

$$H(x) = 1/p(x).$$

If x is a fractional Brownian motion with Hurst exponent H , then $H(x) = H$ a.s.. To estimate the Hurst exponent from a finite sample,

- Compute

$$W_p = \sum_i \frac{|\Delta_{K_i} x|^p}{\sum_{j: [t_{j-1}^L, t_j^L] \cap [t_{i-1}^K, t_i^K] \neq \emptyset} |\Delta_{L_j} x|^p}$$

where the partition π_L is a refinement of π_K ,

- Find \hat{p} such that $W_{\hat{p}} = T$,
- And let $\hat{H} = 1/\hat{p}$.

Realized volatility is rough ($H < \frac{1}{2}$), even if instantaneous volatility is not ($H = \frac{1}{2}$): roughness comes from market microstructure noise.

Error-correcting output codes with ensemble diversity for robust learning in neural networks
Y. Song et al. (2021)

Jointly train (end-to-end) a set of binary classifiers and the ECOC code matrix to combine their outputs, maximizing

- The distance between rows (codewords)
- And the shared information distance between columns (binary tasks), viz the variation of information

$$V(X, Y) = 2H(X \wedge Y) - H(X) - H(Y)$$

to mitigate adversarial attacks.

Time-aware language models as temporal knowledge bases
B. Dhingra et al.

To make your language model aware of time (some facts have an expiration date, e.g., the name of the president), model $P_\theta[\text{mask}|\text{context}, \text{time}]$, e.g., by prefixing each input with the date, “year: 2022”.

Density-preserving data visualization unveils dynamic patterns of single-cell transcriptomic variability
A. Narayam et al. (2020)

With t -SNE and UMAP, the cluster sizes correspond to the number of observations and ignore the density; *denSNE* and *densMAP* augment the objective with a differentiable measure of the total density, to preserve the average distance to the nearest neighbours

$$\frac{\sum_j p_{ij} \|x_i - x_j\|^2}{\sum_j p_{ij}}.$$

Available in `umap-learn`.

Comparing clusterings – an overview
S. Wagner and D. Wagner (2007)

Use an information-based measure, e.g., the normalized mutual information

$$\text{NMI}_1(C_1, C_2) = \frac{I(C_1, C_2)}{\sqrt{H(C_1)H(C_2)}}.$$

The stochastic collocation Monte Carlo sampler: highly efficient sampling from expensive distributions
L.A. Grzelak et al. (2016)

To sample from a distribution F_Y , one could use

$$u \sim \text{Unif}$$

$$y = F_Y^{-1}(u)$$

or, more generally, for an auxilliary distribution F_X (e.g., Gaussian)

$$x \sim F_X$$

$$y = F_Y^{-1}(F_X(x)).$$

If F_Y^{-1} is expensive to compute, stochastic collocation Monte Carlo (SCMC) approximates $F_Y^{-1} \circ F_X$ with *Lagrange polynomials* on *Gauss quadrature points*.

Predicting value at risk for cryptocurrencies using generalized random forests
K. G3rgen et al. (2022)

Random forests can be used for quantile regression: change the splitting criterion to classify the observations wrt the quantile(s) of interest. (Quantile regression forests only use quantile regression in the leaves, without adjusting the splitting criterion.)

A generalized precision matrix for t-Student distributions in portfolio optimization
K. Bax et al. (2022)

With a Gaussian random variable, the *precision matrix* (the inverse of the variance matrix) describes the conditional dependence structure; with non-Gaussian data, this is no longer the case. Alternatives to the precision matrix $\Omega = \Sigma^{-1}$ include:

$$\Omega_{ij} = -\mathbb{E}_x \left[\frac{\partial^2}{\partial x_i \partial x_j} \log f(x) \right]$$

$$\Omega_{ij} = -\mathbb{E}_{x_i, x_j} \left[\mathbb{E}_{x_{\setminus ij}} \left[\frac{\partial^2}{\partial x_i \partial x_j} \log f(x) \right] \mathbf{1}_{x_i^2 + x_j^2 \geq t} \right]$$

$$\Omega_{ij} = \mathbb{E}_x \left[\frac{\partial^2}{\partial x_i \partial x_j} \log f(x) \right]$$

Try them instead of Σ^{-1} in portfolio optimization. [Also check local Gaussian correlation.]

Community detection and portfolio optimization
L. Zhao et al. (2021)

Compute the correlation matrix from daily returns, on a 500-day moving window; build the PMFG (planar maximally filtered graph); compute communities (infomap); pick a stock at random (sic) in each community; build an optimized portfolio from those stocks.

Network diversification for a robust portfolio allocation
M. Jaeger and D. Marinelli

Use the inverse of the degree (or eigen-centrality) on the PMFG as portfolio weights.

LoCoV: low dimension covariance voting algorithm for portfolio optimization
J. Duan and I. Popescu

Compute the optimal portfolio on all subsets of k assets, and average the weights.

Lazy network: a word embedding based temporal financial network to avoid economic shocks in asset pricing models
G. Adosoglou et al. (2022)

- Compute the similarity between the companies' 10K filings for year t ;
- Compute the absolute value of the difference between the similarities in years t and $t - 1$;

- Invest in the periphery of the corresponding graph, i.e., companies with idiosyncratic changes.

A picture is worth a thousand words: measuring investor sentiment by combining machine learning and photos from news
K. Obaid and K. Pukthuanthong

News sentiment can be extracted from news photos; it contains the same information as the sentiment extracted from text. Try a pretrained Inceptionv3, with a head fine-tuned on DeepSent (1000 images) on data from the WSJ: it can predict return and volume at 1 day and 1 week (with different signs).

Realized semibetas: disentangling good and bad downside risks
T. Bollerslev et al.

Consider *four* semibetas, depending on the signs of the market and asset returns. Only $\beta_{\text{market} < 0, \text{asset} > 0}$ and $\beta_{\text{market} < 0, \text{asset} < 0}$ are priced.

The semibetas are defined from the decomposition

$$\beta = \frac{\text{Cov}(r_{\text{market}}, r)}{\text{Var } r_{\text{market}}}$$

$$= \frac{c_{++} + c_{--} - c_{+-} - c_{-+}}{\text{Var } r_{\text{market}}}$$

$$= \beta_{++} + \beta_{--} - \beta_{+-} - \beta_{-+}.$$

Investor attention and stock returns
J. Chen et al. (2020)

Combine 12 proxies of investor attention (aggregated over all stocks):

- Abnormal trading volume (1m vs 12m);
- Extreme returns (1m vs 12m);
- Past returns (12m);
- Nearness to the Dow 52-week high, and historical high;
- Analyst coverage;
- Change in advertising expenses;
- Mutual fund inflow, and outflow;
- Media coverage;
- Google search volume;
- Edgar search volume.

Aggregate with

- PCA,
- *scaled PCA* (sPCA)

$$\beta_i = \beta(y \sim x_i)$$

$$A = \text{PC}_1(\beta_1 x_1, \dots, \beta_n x_n)$$

y : future returns

x : proxies

- or *partial least squares* (PLS)

$$\pi_i = \beta(x_1 \sim y) \quad \text{time series regressions}$$

$$A_t = \beta(x_t \sim \pi) \quad \text{cross-sectional regressions.}$$

***Some applications of TDA
on financial markets***
M.A. Ruiz-Ortiz et al. (2022)

Using

- Daily returns of 4 indices, for 15 days (15 points in 4-dimensional space),
- or 4-dimensional delay embeddings of cryptocurrency returns, for 50 days (50 points)

compute the persistence landscape λ_t and $\text{Var} \|\lambda_t\|_1$ or $\|\lambda_t\|_1 + \|\lambda_t\|_1 - \|\lambda_{t-1}\|_1$ to detect crashes.

The persistent homology turbulence index (PHTI) is obtained by

- Taking 60 days of industry log-returns (60 points in k -dimensional space), or a delay embedding of a single time series;
- Computing the persistence diagram P_t , then the Wasserstein distances between P_t and P_{t-1} , then a 60-day moving average, then the quantile over a 5-year window,
- or, alternatively, by computing the persistence landscape and $\|\lambda_{t-\tau} - \lambda_t\|_2$.

***Tail-GAN: non-parametric scenario
generation for tail risk estimation***
R. Cont et al. (2022)

To generate scenarios to compute the VaR or the ES, train a GAN, whose discriminator evaluates the quality of VaR and ES estimates (sorting is a.e. differentiable) using a *scoring function* (the pair (VaR, ES) is *elicitable* – the formula is complicated) on a set of static and *dynamic* trading strategies.

***Sensitivity measures
based on scoring functions***
T. Fissler and S.M. Pesenti (2022)

A statistic T is *elicitable* if there is a scoring function S such that $T(X) = \text{Argmin}_t E[S(t, X)]$. For instance, mean $S(\mu, x) = (\mu - x)^2$, median $S(m, x) = |m - x|$, VaR_α , or the pair $(\text{VaR}_\alpha, \text{ES}_\alpha)$ are elicitable; for each of those, there is actually a (known) family of scoring functions.

The sensitivity of T wrt information Y is

$$\frac{E[S] - E[S|Y]}{E[S]}.$$

***Multi-objective reward generalization:
improving performance of deep reinforcement
learning for selected applications
in stock and cryptocurrency trading***
S. Cornalba and C. Disselkamp

If the reward is a vector rather than a scalar (e.g., $(\mu, \mu/\sigma)$), optimize a random positive linear combination of rewards, resampled at each step.

***Weisfeiler and Lehman go cellular:
CW networks***

The Weisfeiler-Lehman isomorphism test can be generalized to simplicial of CW complexes (*i.e.*, cells with arbitrary shapes). A cell has 4 types of neighbours:

- Boundary adjacent: cells on the boundary, one dimension smaller;
- Coboundary adjacent: cells on the boundary, one dimension larger;
- Lower adjacent cells: cells of the same dimension, sharing a lower-dimensional cell on their boundary;
- Upper adjacent cells: cells of the same dimension, on the boundary of the same higher dimensional cell.

Coboundary and lower adjacencies can be omitted without affecting the expressive power of the test.

A graph can be lifted to a CW complex in many ways, e.g., the *clique complex* (each $(k+1)$ -clique becomes a k -cell), limited to cliques of size at most n , or the *cycle complex*.

Message-passing GNNs can be generalized to those lifted graphs.

***A reverse expected shortfall
optimization formula***
Y. Guan et al. (2022)

To minimize the expected shortfall

$$\begin{aligned} \text{ES}_\alpha &= \frac{1}{1-\alpha} \int_\alpha^1 \text{VaR}_\beta(X) d\beta \\ &= \min_{t \in \mathbf{R}} t + \frac{1}{1-\alpha} E[(X-t)_+], \end{aligned}$$

it suffices to minimize a linear function plus the excess mean loss (for each t).

Conversely,

$$E[(X-t)_+] = \max_{\alpha \in [0,1]} (1-\alpha) [\text{ES}_\alpha(X) - t]$$

is useful in worst-case (*i.e.*, robust) optimization.

These formulas generalize to optimized certainty equivalent (OCE) risk measures of the form

$$R[X] = \int_{t \in \mathbf{R}} t + \frac{1}{\beta} E[v(X-t)].$$

***Improved iterative methods
for solving risk parity portfolio***
J. Choi and R. Chen (2022)

***Doubly truncated moment risk measures
for elliptical distributions***
B. Zuo and C. Yin (2022)

The *tail conditional moments* are

$$\begin{aligned} \text{TCM}_{q,n}(X) &= E[(X - \text{TCE}_q X)^n \mid X > x_q] \\ \text{TCE}_q(X) &= E[X \mid X > x_q] \\ x_q &= q\text{-th quantile.} \end{aligned}$$

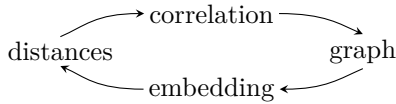
For the doubly truncated moments, replace the condition $X > x_q$ with $x_p < X < x_q$.

Multivariate doubly truncated moments for generalized skew-elliptical distributions with applications to multivariate tail conditional risk measures
B. Zuo and C. Yin (2022)

For many multivariate distributions, one can compute $E[Y|a \leq Y \leq b]$ and $E[YY'|a \leq Y \leq b]$.

RPS: portfolio asset selection using graph-based representation learning
M.A. Fazli et al.

In portfolio optimization, replace the correlation matrix with a similarity matrix computed from the distance matrix of a node embedding (Node2Vec) computed from the complete graph with correlations as weights.



Connecting Sharpe ratio and Student t -statistic, and beyond
E. Benhamou

If $X \sim N(\mu, \sigma^2)$, the $\widehat{SR} = \hat{\mu}/\hat{\sigma}$ is (non-centered) Student.

The statistics of Sharpe ratios
A.W. Lo (2002)

For iid returns,

$$\sigma(\widehat{SR}) = \sqrt{\frac{1 + \frac{1}{2}SR}{T}}.$$

For non-iid returns, annual and monthly Sharpe ratios are related by ($q = 12$)

$$\frac{SR(q)}{SR} = \frac{q}{\sqrt{q + 2 \sum_{k=1}^{q-1} (q-k)\rho_k}}.$$

In particular, for AR(1) returns,

$$\frac{SR(q)}{SR} = \sqrt{q} \left[1 + \frac{2\rho}{1-\rho} \left(1 - \frac{1-\rho^q}{q(1-\rho)} \right) \right]^{-1/2}.$$

The variance of \widehat{SR} increases by

$$\frac{\text{Var } \widehat{SR}(q)}{\text{Var } \widehat{SR}} = 1 + \frac{2 \sum_{j=1}^{q-1} (1-j/q)^2}{1 + 2/SR^2}.$$

Multi-portfolio internal rebalancing processes
K. Francis-Staite (2022)

The (internal) rebalancing problem

i : asset class

j : portfolio

a_i : value of asset class i

p_j : value of portfolio j

M_{ij} : desired proportion of asset class i in portfolio j

where $\sum_i M_{ij} = 1$, $\sum a_i = \sum p_j$, looks for A , as close to M as possible (in some sense), such that $\sum_i A_{ij} = 1$ and $Ap = a$ (one may add a constraint $\forall ij \ M_{ij} = 0 \Rightarrow A_{ij} = 0$).

In *banker rebalancing*, all the portfolios except one (the banker) are given their target allocation; the banker receives the remaining funds. In *linear rebalancing*, each asset class is considered separately, and the over-/under-weights are distributed equally (not in proportion of the weight of the portfolio). *Market-invariant rebalancing* is more equitable.

Modeling bid and ask price dynamics with an extended Hawkes process and its empirical applications for high-frequency stock market data
K. Lee and B.K. Seo

Four-dimensional Hawkes process, for the up and down bid and ask movements

$$\lambda_t = \mu_t + \int_{-\infty}^t e^{-\beta(t-u)} A_u dN_u$$

where μ_t and A_t are observed (or $A \in \mathbf{R}^{4 \times 4}$ constant).

Multi-asset spot and option market simulation
M. Wiese et al. (2021)

Do not directly generate a grid of call prices (for various expiries and strikes) but a grid of *discrete local volatilities* (DLV)

$$\sigma_{tk} = \sqrt{\frac{\theta_{tk}}{\frac{1}{2}k^2 \Delta t \Gamma_{tk}}}$$

$$\theta_{tk} = C_{t+\Delta t, k} - C_{t, k} \approx \partial_t C_{tk}$$

$$\Gamma_{tk} \approx \partial_{kk} C_{tk}$$

to enforce the “no static arbitrage” constraint.

Reduce the dimension of the DLV grid (from 26 to 3) with an auto-encoder $\sigma \mapsto x \mapsto \sigma$.

Train a conditional model $x_{t+1}|x_t, x_{t-1}$ with a normalizing flow

$$u_{t+1} \sim U(0, 1)^3$$

$$x_{t+1} \leftarrow f(u_{t+1}; x_t, x_{t-1}).$$

The result still has a drift: it is not a martingale.

For several assets, sample u_{t+1} from a Gaussian copula with a block structure (or a non-parametric, flow-based copula).

Risk-neutral market simulation **M. Wiese and P. Murray (2022)**

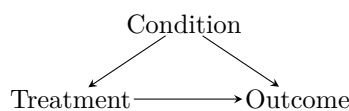
Remove the drift by reweighting the samples (the weights are the solution of a convex optimization problem).

Lazy prices **L. Cohen et al (2019)**

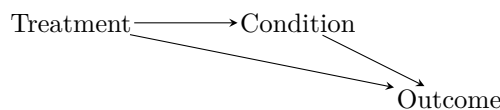
Changes in 10K reports have a (negative) predictive power (earnings, profitability, news, bankruptcies), especially the “risk factor” section and statements about the executive team (CEO, CFO) and litigation.

Introduction to causal inference **B. Neal (2020)**

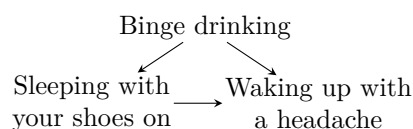
1. Simpson’s paradox shows that omitting important variables can lead to the opposite conclusion: for instance, if two treatments are available for some medical condition, a non-invasive and an invasive one, the former used mostly for mild conditions and the latter for more severe ones, the invasive one can appear worse on the whole population, even though it is better on both sub-populations, simply because it tends to be used mostly on difficult patients.



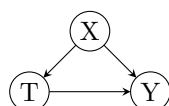
But a different causal graph, e.g.,



(if the invasive treatment is administered with delays, letting the patient’s condition worsen), the same numbers lead to the opposite conclusion.



2. The **potential outcomes** $Y_i(0)$ and $Y_i(1)$ are the values of the outcome Y for subject i if it does not ($T = 0$) or does ($T = 1$) receive the treatment.



We only observe one of them,

i	T	Y	$Y(1)$	$Y(0)$	$Y(1) - Y(0)$
1	0	0	?	0	?
2	1	1	1	?	?
3	1	0	0	?	?
4	0	0	?	0	?
5	0	1	?	1	?
6	1	1	1	?	?

and we are interested in their differences, the *individual treatment effect*

$$\text{ITE}_i = Y_i(1) - Y_i(0)$$

or the *average treatment effect*

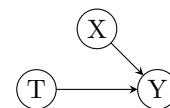
$$\text{ATE}_i = E[Y_i(1) - Y_i(0)].$$

In general, it is different from the *associational difference*, which is easy to compute.

$$\text{ATE} \neq E[Y|T = 1] - E[Y|T = 0]$$

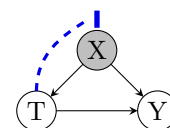
They are equal if the *ignorability* (or *exchangeability*) assumption is satisfied:

$$Y(0), Y(1) \perp\!\!\!\perp T.$$



(We can ignore how the subjects were selected for the treatment; the treated and the control groups are comparable). This is the case for *randomized control trials* (RCT). More generally, if the *unconfoundedness* assumption is satisfied

$$Y(0), Y(1) \perp\!\!\!\perp T | X,$$



we can compute the ATE with the **adjustment formula**

$$\begin{aligned} \text{ATE} &= E[Y(1) - Y(0)] \\ &= E_X [E[Y | T = 1, X] - E[Y | T = 0, X]]. \end{aligned}$$

A few more assumptions are actually needed:

- Unconfoundedness;
- Positivity (overlap): the supports of $X | T = 0$ and $X | T = 1$ are the same, *i.e.*, all subgroups of data appear both in $[T = 1]$ and $[T = 0]$;
- Consistency: no multiple versions of the treatment;
- No interference: $Y_i(t_i)$ does not depend on $t_j, j \neq i$.

Causal inference has two steps:

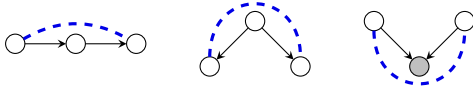
- *Identification*, *i.e.*, converting a *causal estimand* (e.g., the ATE) into a *statistical estimand* (e.g., the adjustment formula);
- *Estimation*, *i.e.*, computing the statistical estimand from the data (this often involves some statistical or machine learning model).

3. A *Bayesian network* is a DAG encoding a joint probability distribution as

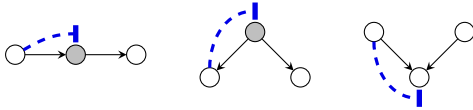
$$P(x_1, \dots, x_n) = \prod_i P(x_i | \text{pa}_i).$$

In particular, X_i is independent of its non-descendants given its parents (*local Markov assumption*). If adjacent nodes are dependent, the DAG is *minimal*.

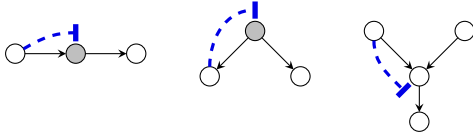
Association flows along *chains*, *forks* and conditioned **immoralities** (*colliders*, *v-structures*)



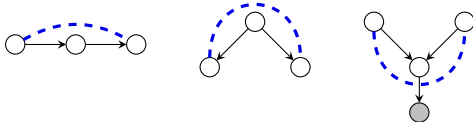
Association is blocked by conditioned chains, conditioned forks, unconditioned immoralities (for immoralities, we need to condition, or avoid conditioning, on any descendant of the immorality).



Blocked paths contain one of

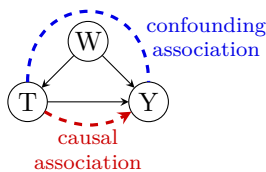


Unblocked paths only contain



Two nodes are **d-separated** if all paths between them are blocked; *d*-separation implies conditional independence (global Markov property).

$$X \perp\!\!\!\perp_G Y | Z \implies X \perp\!\!\!\perp_P Y | Z$$



4. Causal quantities, such as $P[Y|\text{do}(T = t)]$, can be computed

- From the **structural causal model** (SCM) by replacing the equation for T with $T = t$;
- From the Bayesian network factorization, by setting the factor for T to $\mathbf{1}_{T=t}$;
- From the causal graph, by cutting all edges into T (**manipulated graph**).

It is sometimes possible to convert a causal quantity (with the “do” operator) into a statistical one, *i.e.*, computable from observational data (identification). For instance, if W satisfies the **backdoor criterion** $T \rightarrow Y$, *i.e.*,

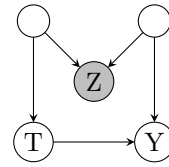
- W blocks all backdoor (*i.e.*, non-causal) paths from T to Y ;
- W does not contain any descendant of T ,

then, W is a **sufficient adjustment set**:

$$P[Y|\text{do}(T = t)] = \sum_w P[Y, t, w]P[w]$$

(*backdoor adjustment*).

Beware of *M-bias*: in the following graph, do not condition on Z – it opens a backdoor path.



5. In **randomized experiments**, association is causation. This can be seen by:

- Covariate balance: the treatment and the control groups are identical

$$P[X | T = 1] \stackrel{d}{=} P[X | T = 0]$$

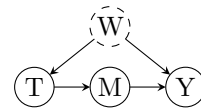
- Exchangeability (unconfoundedness)

$$E[Y(1) | T = 1] = E[Y(1) | T = 0]$$

$$E[Y(0) | T = 0] = E[Y(0) | T = 1]$$

- Absence of backdoor path, because T has no incoming edges: $W = \emptyset$ is a sufficient adjustment set.

6. Backdoor adjustment is sufficient but not necessary for identifiability. For instance, in presence of unobserved confounders W ,



we may be able to use the *frontdoor adjustment*, by separately computing the causal effects $T \rightarrow M$ and $M \rightarrow Y$, and combining them.

$$P[Y|\text{do}(t)] = \sum_m P[m|t] \sum_t P[y|m, t']P[t'].$$

The **do-calculus** gives a complete list of (3) transformations to identify causal quantities, whenever possible.

7. To estimate the ATE with a sufficient adjustment set W ,

$$\begin{aligned} \tau &= E[Y(1) - Y(0)] \\ &= E_W[E[Y | T = 1, W] - E[Y | T = 0, W]] \end{aligned}$$

we can fit a statistical model to

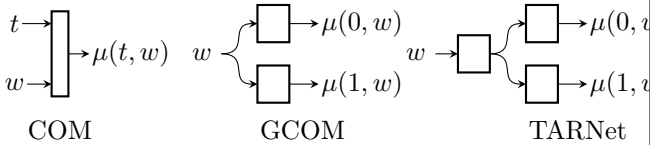
$$\mu(t, w) = E[Y | T = t, W = w]$$

(conditional outcome modeling, COM), or separate models to

$$\mu_0(w) = E[Y | T = 0, W = w]$$

$$\mu_1(w) = E[Y | T = 1, W = w]$$

(grouped conditional outcome modeling, GCOM) or with a neural net learning a representation of w (TAR-Net).



The X -learner approach:

- Fits models $\hat{\mu}_0(x)$, $\hat{\mu}_1(x)$;
- Fits models

$$\hat{\tau}_1(x) = Y(1) - \hat{\mu}_0(x),$$

$$\hat{\tau}_0(x) = \hat{\mu}_1(x) - Y(0);$$

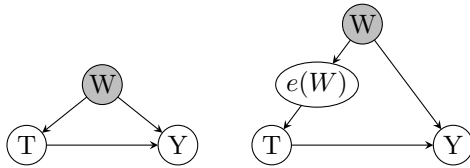
- Combines them

$$\hat{g}(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)$$

for some $g : \mathcal{X} \rightarrow [0, 1]$, e.g., the propensity score, or 1/2, or 0, or 1.

The conditioning set W can be high-dimensional, but it can be replaced with the (1-dimensional) *propensity score* $e(w) = P[T = 1 | W = w]$:

$$Y(0), Y(1) \perp\!\!\!\perp W \implies Y(0), Y(1) \perp\!\!\!\perp e(W).$$



In particular,

$$E[Y(t)] = E \left[\frac{\mathbf{1}_{T=t} \cdot Y}{P[t|W]} \right].$$

The propensity score is unknown, but can be estimated, leading to the *inverse propensity weighting* (IPW) estimator

$$\hat{\tau} = \text{Mean}_{i: t_i=1} \frac{y_i}{\hat{e}(w_i)} - \text{Mean}_{i: t_i=0} \frac{y_i}{1 - \hat{e}(w_i)}$$

(to trade variance for bias, trim the propensity scores \hat{e} to $[\varepsilon, 1 - \varepsilon]$).

Other methods include:

- Doubly robust methods, which model both $\mu(t, w)$ and $e(w)$;
- Matching;

- Double machine learning:

$$\text{res}(Y \sim W) \sim \text{res}(T \sim W);$$

- Causal trees and forests.

For confidence intervals, use the bootstrap.

8. The *observational-counterfactual decomposition* of the ATE

$$\tau = E[T(1) - Y(0)]$$

$$= P[T = 1] E[Y|T = 1] + P[T = 0] E[Y(1)|T = 0]$$

$$- P[T = 1] E[Y(0)|T = 1] + P[T = 0] E[Y|T = 0]$$

gives **bounds** on τ , (rather large but) finer if we make more assumptions, such as:

- Monotone treatment response: $\forall i Y_i(1) \geq Y_i(0)$;
- Monotone treatment selection

$$E[Y(1)|T = 1] \geq E[Y(1)|T = 0]$$

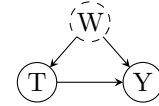
$$E[Y(0)|T = 1] \geq E[Y(0)|T = 0];$$

- Optimal treatment selection

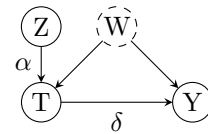
$$Y_i = 1 \implies Y_i(1) \geq Y_i(0)$$

$$Y_i = 0 \implies Y_i(0) \geq Y_i(1).$$

Sensitivity analysis examines the question “how strong should the effect of an unobserved confounder W on T and Y be for the causal effect of T on Y to drop below some threshold” (as a contour plot, causal effect \sim confounding effects).



7. Instrumental variables allow identification in presence of unobserved confounders, but they require parametric assumptions.



For instance, if the structural equations are linear, we can measure α and $\alpha\delta$ and use the *Wald estimand*

$$\delta = \frac{E[Y | Z = 1] - E[Y | Z = 0]}{E[T | Z = 1] - E[T | Z = 0]}.$$

In the continuous linear setting,

$$\delta = \frac{\text{Cov}(Y, Z)}{\text{Cov}(T, Z)};$$

it can also be estimated with *2-stage least squares* (2sls)

$$\text{lm}(Y \sim \text{predict}(T \sim Z)).$$

If the instrument is an encouragement to take the treatment, we can stratify the population into

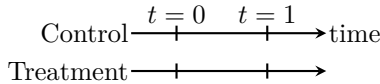
- Compliers: $T = Z$;

- Always-takers: $T = 1$;
- Always deniers: $T = 0$;
- Defiers: $T = \neg Z$.

If there are no defiers (*monotonicity*), we can compute the *complier average causal effect*, non-parametrically, still with the Wald estimand.

$$\begin{aligned} \text{CACE} &= E[Y(1) - Y(0) | T(1) = 1, T(0) = 0] \\ &= \frac{E[Y | Z = 1] - E[Y | Z = 0]}{E[T | Z = 1] - E[T | Z = 0]} \end{aligned}$$

10. Adding a time dimension



and making the *parallel trends assumptions*

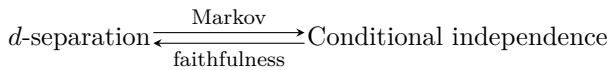
$$Y_1(0) - Y_0(0) \perp\!\!\!\perp T,$$

the *average treatment effect on the treated* (ATT) can be estimated with the **difference-in-differences**

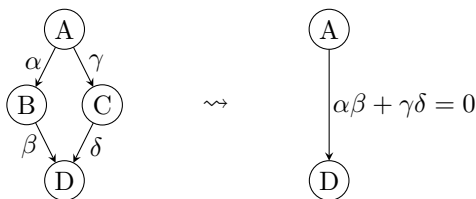
$$\begin{aligned} \text{ATT} &= E[Y_1(1) - Y_0(0) | T = 1] \\ &= E[Y_1 - Y_0 | T = 1] - E[Y_1 - Y_0 | T = 0]. \end{aligned}$$

(But the parallel trends assumption is not stable under data transformations, e.g., log.)

11. Conditional independence and d -separation are equivalent if the graph is Markov and *faithful* to the probability distribution.



Faithfulness is not always possible: some effects could cancel out.



Two graphs are *Markov equivalent* iff they have the same *skeleton* and *immoralities*.

The **PC algorithm** identifies the Markov equivalence class of a probability distribution:

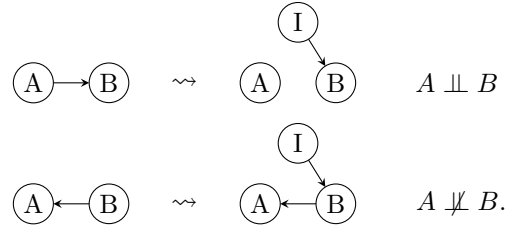
- Find the skeleton, with conditional independence tests;
- Find the immoralities, with conditional independence tests;
- Orient the remaining edges, whenever possible, assuming we have detected all the immoralities.

In the linear Gaussian setting, we cannot distinguish between $A \rightarrow B$ and $A \leftarrow B$, but in the

- Linear, non-Gaussian noise,
- Or nonlinear, additive noise

settings, we can.

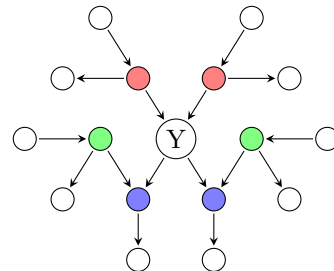
12. While it is not possible to identify the essential graph $A \rightarrow B$, it is possible with **interventions**. For instance, an intervention on B gives



For $n \geq 3$, $n - 1$ 1-node interventions are sufficient. For multinode interventions, $\lceil \log_2 n \rceil + 1$, or $\lceil \log_2 c \rceil$ are sufficient (and necessary, in the worst case), where c is the size of the largest clique, if you already know the essential graph.

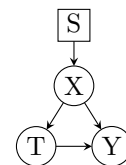
There are similar results for parametric interventions, *i.e.*, if we change $P[X_i | \text{pa}_i]$ instead of removing pa_i .

13. Covariate shift refers to the situation where the distribution of (x, y) changes between training and testing, but that of $y|x$ remains. More generally, let us assume that the distribution of x changes e.g., through some (possibly unknown) intervention, but the causal structure remains. Without intervention, we can predict Y from its *Markov blanket* (parents, children, lovers),



but with intervention, we should only use the *causal parents* of Y .

To study the *transportability* of causal effects across populations, add a “selection node” to indicate which nodes have their causal mechanism changed.



14. Counterfactuals, such as $P[Y(t) | T = t', Y = y]$, can be computed from a structural causal model, but they cannot be written with the do operation, or obtained from observations.

For instance, $Y := UT + (1 - U)(1 - T)$, where

- $Y = 1$ happy
- $T = 1$ get a dog
- $U = 1$ dog person.

If we observe $T = 0$, $Y = 0$, we can compute U (*abduction*), modify the SCM by setting $T := t$, and compute $Y(t)$. If $Y = f(U)$ is not invertible, use a prior on U to get a *probabilistic counterfactual*.

While unit-level counterfactuals require an SCM, *population-level counterfactuals* $E[Y(t)|T = t']$ can be computed as the CATE (“potential outcome calculus” generalizes do-calculus).

Causal inference: what if
M.A. Hernán and J.M. Robins (2020)

***DoWhy: an end-to-end library
for causal inference***
A. Sharma and E. Kiciman

Causal inference requires 4 steps:

- Provide the causal graph and specify the “treatment” and “outcome” variables;
- Identify the ATE (average treatment effect), *i.e.*, convert the causal estimand $E[Y|\text{do}(X = x)]$ into a statistical estimand (*i.e.*, something computable from observational data): backdoor criterion, front-door criterion, instrumental variables, etc.
- Compute the ATE (propensity score, linear regression, 2-stage least squares, etc.)
- Robustness tests:
 - Add a random variable;
 - Replace the treatment with a random variable (placebo), or the outcome;
 - Use a subset of the data;
 - Add an unmeasured common cause.

***Approximate kernel-based
conditional independence tests
for fast non-parametric causal discovery***
E.V. Strobl and S. Visweswaran

Replace X, Y, Z with random Fourier features and define $\Sigma_{XY \cdot Z} = \Sigma_{XY} - \Sigma_{XZ}\Sigma_{ZZ}^{-1}\Sigma_{ZY}$; then $\Sigma_{XY \cdot Z} = 0$ iff $X \perp\!\!\!\perp Y|Z$.

R implementation in RCIT.

***Learning functional causal models
with generative neural networks***
O. Goudet et al.

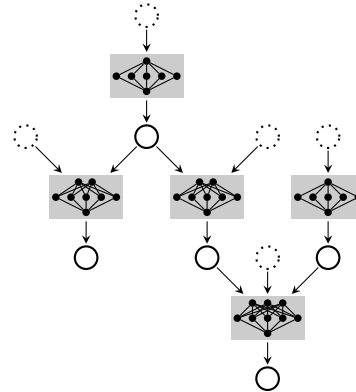
The causal generative neural network (CGNN) builds a functional causal model (FCM) by combining

- A global approach, selecting a (small number of) CPDAG(s)
- And a local approach (explaining the asymmetry between $Y = f(X)$ and $X = f(Y)$ – one is often simpler than the other), fitting models of the form $X_i = f(\text{pa}_i, E_i)$ where f is a shallow network and the E_i ’s are independent noises selecting the best generative model by looking at the maximum mean

discrepancy (MMD)

$$\text{MMD}(\mathcal{D}, \hat{\mathcal{D}}) = \frac{1}{n^2} \sum_{ij} k(x_i, x_j) + \frac{1}{n^2} \sum_{ij} k(\hat{x}_i, \hat{x}_j) - \frac{1}{n^2} \sum_{ij} k(x_i, \hat{x}_j)$$

for some kernel k (e.g., Gaussian).



***Structural agnostic modeling:
adversarial learning of causal graphs***
D. Kalainathan et al.

For each variable X_i , train a neural network $X_i = f(a_i \odot X_{\setminus i}, E_i)$, where the a_i ’s are binary gates (with penalties to ensure sparsity and acyclicity).

***Efficient differentiable quadratic programming
layers: an ADMM approach***
A. Butler and R.H. Kwon (2021)

Quadratic optimization problems

$$\begin{array}{ll} \text{Find} & z \\ \text{To minimize} & \frac{1}{2} z' Q z + z' p \\ \text{Such that} & A z = b \\ & \ell \leq z \leq u \end{array}$$

can be solved with ADMM

$$\begin{aligned} f(x) &= \frac{1}{2} z' Q z + z' p + I_{Ax=b} \\ g(z) &= I_{\ell \leq z \leq u} \end{aligned}$$

$$\begin{array}{ll} \text{Find} & x, z \\ \text{To minimize} & f(x) + g(z) \\ \text{Such that} & x = z; \end{array}$$

each step can be solved analytically.

$$\begin{aligned} x &\leftarrow \underset{x: Ax=b}{\text{Argmin}} \frac{1}{2} x' Q x + x' p + \frac{\rho}{2} \|x - z + \mu\|^2 \\ z &\leftarrow \underset{\ell \leq z \leq u}{\text{Argmin}} \|x - z + \mu\|^2 \\ \mu &\leftarrow \mu + x - z \end{aligned}$$

It can also be used for optimization as a layer (when Q, p, A, b, ℓ, u depend on θ): to compute the gradient, use the fixed point property of the ADMM iteration: it is less computationally-intensity than the KKT conditions.

1. Machine learning on graphs covers the following tasks:

- Node classification,
- Relation prediction,
- Graph classification (or regression)

but also: node clustering (community detection), graph clustering, etc.

2. Traditional node and graph features include: degree, centralities (eigenvector, closeness, betweenness), clustering coefficient (different normalizations), graphlet count, Weisfeiler-Lehman kernel, path-based kernel (e.g., distribution of degree sequences in random walks, or in shortest paths).

Measures of neighbourhood overlap include:

$$\frac{|N(u) \cap N(v)|}{d_u + d_v}, \quad \frac{|N(u) \cap N(v)|}{\sqrt{d_u d_v}}, \quad \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

$$\sum_{w \in N(u) \cap N(v)} \frac{1}{d_w}, \quad \sum_w \frac{1}{\log d_w}, \quad \sum \beta^i (A^i)_{uv};$$

some can be directly computed from the adjacency matrix:

$$(I - \beta A)^{-1} - I, \quad D^{-1}(I - \beta A)^{-1}D^{-1}.$$

The graph Laplacian $L = D - A$ satisfies

$$x' L x = \sum_{u,v \in E} (x_u - x_v)^2;$$

it is often normalized as $L_{\text{sym}} = D^{-1/2} L D^{1/2}$ or $L_{\text{rw}} = D^{-1} L$. The *ratio cut* and *normalized cut*

$$\text{RCut}(\mathcal{A}_1, \dots, \mathcal{A}_n) = \sum_k \frac{|(u, v) \in E : u \in \mathcal{A}_k, v \notin \mathcal{A}_k|}{|\mathcal{A}_k|}$$

$$\text{NCut}(\mathcal{A}_1, \dots, \mathcal{A}_n) = \sum_k \frac{|(u, v) \in E : u \in \mathcal{A}_k, v \notin \mathcal{A}_k|}{\sum_{u \in \mathcal{A}_k} d_u}$$

are maximized (after relaxation) by the second smallest eigenvector of L or L_{rw} . Spectral clustering applies k -means to the smallest eigenvectors of L .

3. We can compute node embedding with an encoder-decoder architecture (for a fixed graph)

$$\begin{aligned} \text{ENC} : \quad & V \longrightarrow \mathbf{R}^d \\ \text{DEC} : \quad & \mathbf{R}^d \times \mathbf{R}^d \longrightarrow \mathbf{R}^+. \end{aligned}$$

The decoder could be $(a, b) \mapsto \|a - b\|^2$, $a'b$, or $e^{a'b}$; the loss could be $\text{DEC}(z_u, z_v) \cdot S_{u,v}$, $\|\text{DEC}(z_u, z_v) - S_{u,v}\|$, $-S_{u,v} \log \text{DEC}(z_u, z_v)$ for some node similarity measure S , e.g., $S = A$, or $S = (A, A^2, \dots, A^k)$, or some node neighbourhood overlap measure.

These can often be seen as matrix factorizations, e.g.,

$$\underset{Z}{\text{Minimize}} \|ZZ' - S\|_2^2,$$

or random walk embeddings.

Those shallow embeddings have limitations.

4. These decoders can be generalized to multirelational graphs, e.g., knowledge graphs (KG), and matrix factorizations become tensor factorizations:

$$\begin{aligned} & z'_u R_\tau z_v \\ & - \|z_u + r_\tau - z_v\| \\ & - \|g_{1,\tau}(z_u) + r_\tau = g_{2,\tau}(z_v)\| \\ \langle z_u, r_\tau, z_v \rangle &= \sum_i z_{ui} r_{\tau i} z_{vi} \\ \text{Re} \langle z_u, r_\tau, \bar{z}_v \rangle & \\ & - \|z_u \odot r_\tau = z_v\| \end{aligned}$$

(the complex decoder allows for asymmetric relations).

5. GNNs are neural networks transforming the adjacency matrix of a graph in a permutation-equivariant (or -invariant) way: $f(PAP') = Pf(A)$ (or $f(PAP') = f(A)$), by message passing:

$$\begin{aligned} m_{N(u)} &\leftarrow \text{AGGREGATE}(\{h_v, v \in N(u)\}) \\ h_u &\leftarrow \text{UPDATE}(h_u, m_{N(u)}). \end{aligned}$$

It is common to add self-loops and omit the explicit update step:

$$h_u \leftarrow \text{AGGREGATE}(\{h_v, v \in N(u) \cup \{u\}\}).$$

The aggregation can be

$$m_u = \sum_v h_v \quad (\text{GNN})$$

$$m_u = \frac{1}{|N(u)|} \sum_v h_v$$

$$m_u = \sum_v \frac{h_v}{\sqrt{|N(u)||N(v)|}} \quad (\text{GCN}).$$

The sum can be replaced by

$$m_u = \text{MLP}\left(\sum_{v \in N(u)} \text{MLP}(h_v)\right)$$

$$m_u = \text{MLP}\left(\frac{1}{|N(u)|!} \sum_{\pi \in \mathfrak{S}_{|N(u)|}} \text{LSTM}(h_{v_1}, \dots, h_{v_{|N(u)|}})\right)$$

(approximated with a random set of permutations, or even a single one, corresponding to some canonical ordering, e.g., by degree).

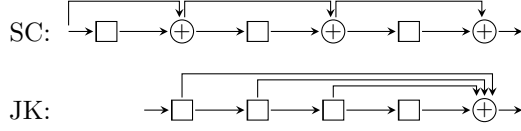
We can also add attention

$$\begin{aligned} m_{N(u)} &= \sum_v \alpha_{u,v} h_v \\ \alpha_{u,v} &\propto \exp a'[Wh_u \oplus Wh_v] \quad (\text{GAT}) \\ \alpha_{u,v} &\propto \exp(h'uWh_v) \quad (\text{bilinear attention}) \\ \alpha_{u,v} &\propto \exp \text{MLP}(h_u, h_v). \end{aligned}$$

To limit oversmoothing, try skip connections, gated updates,

$$h_u \leftarrow \text{GRU}(h_u, m_{N(u)})$$

or jumping knowledge (JK) connections.



GNNs can be generalized to multirelational graphs, *i.e.*, graph with several edge types (or edge features).

Graph pooling is needed for graph-level predictions:

- Sum or mean aggregation of the node embeddings
- Attention with LSTM:

$$\begin{aligned} q &\leftarrow \text{LSTM}(o, q) \\ e_v &\leftarrow f(z_v, q) \\ a_v &\propto \exp e_v \\ o &= \sum a_v z_v \end{aligned}$$

- Graph coarsening

$$\begin{aligned} A &\leftarrow S'AS \\ X &\leftarrow S'X \end{aligned}$$

for some cluster assignment matrix S .

6. Pretraining with neighbourhood reconstruction loss does not work well. Try to maximize the mutual information between node embeddings and graph embeddings:

$$- \mathbb{E}_{\text{data}} \log D(z_u, z_G) - \mathbb{E}_{\text{corrupted data}} \log(1 - D(\tilde{z}_u, z_G)).$$

For deep networks, use gated updates.

7. A GNN layer is a convolution: it can be expressed without message passing, with the adjacency matrix A , or $A_{\text{sym}} = D^{-1/2}AD^{-1/2}$, or some Laplacian, $L = D - A$, $L_{\text{sym}} = D^{-1/2}LD^{-1/2}$

$$Z = \text{MLP}(f(A) \text{MLP}(X))$$

with, *e.g.*,

$$\begin{aligned} f(A) &= \sum (I - \alpha \tilde{A})^k \\ \tilde{A} &= (D + I)^{-1/2}(A + I)(D + I)^{-1/2}. \end{aligned}$$

GNNs reproduce the operations in the Weisfieler-Lehman graph isomorphism test (which cannot distinguish between \square and ∇ – this is a strong limitation, and breaking it is an open problem).

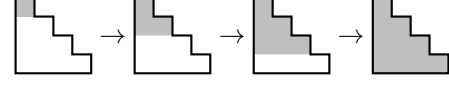
9. Variational graph autoencoders generate graphs from node embeddings with a dot product decoder:

$$\begin{aligned} \mu, \sigma &= \text{GNN}(A, X) \\ z &\sim N(\mu, \sigma) \\ P[A_{ij} = 1] &= \sigma(z'_i z_j). \end{aligned}$$

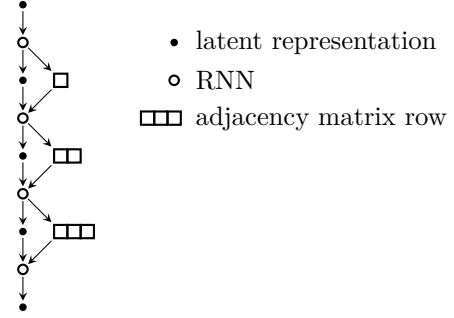
They are good for reconstruction, but not for graph generation. A graph-level VAE uses a graph (not node) embedding and generates an adjacency matrix in one

go: $A = \sigma(\text{MLP}(z_G))$; the nodes of the generated graph need to be matched to the original ones, *e.g.*, with a heuristic ordering.

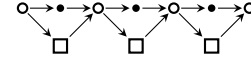
Autoregressive decoders generate the rows (of the lower triangular part) of A one by one, each conditioned on the previous ones, as with the preferential attachment model



with a graph-level RNN



where each row $\circ \rightarrow \square$ is generated by a node-level RNN



The graph recurrent attention network (GRAN) also generates the rows one by one, but each new row is computed by a GNN on the graph built so far.

NodePiece:
compositional and parameter-efficient representations of large knowledge graphs
M. Galkin et al.

Node embeddings do not scale to very large networks: we would have to learn and store the embedding of all the nodes. Instead, select (a smaller number of) “anchor” nodes (at random), whose preliminary embeddings will be learned, and compute the embeddings of all the nodes on the fly, with a (learned) MLP, taking as inputs the preliminary embeddings of the k closest anchors, their distances (numbers of hops), and the features of the edges to/from that node.

Graphon pooling in graph neural networks
A. Parada-Mayorga et al. (2020)

Pooling, in graph neural networks (GNN), is often deterministic and arbitrary. Instead, consider the input graph as a locally constant graphon and sample from it, to have increasingly coarser approximations of the input graph.

There are two ways of sampling from a graphon $W : [0, 1]^2 \rightarrow [0, 1]$: either pick points at random $x_i \sim U(0, 1)$ and link them with probabilities $W(x_i, x_j)$, or randomly partition the interval $[0, 1]$ into $\sqcup U_i$ and link nodes i and j with probability $\iint_{U_i \times U_j} W(x, y) dx dy / |U_i \times U_j|$; use the latter.

**DDG-DA: data distribution generation
for predictable concept drift adaptation**
W. Li et al. (2022)

To deal with slow concept drift, forecast the evolution of the data distribution to adapt the model.

**Conditional Sig-Wasserstein GANs
for time series generation**
H. Ni

By analogy with the Wasserstein distance

$$W_1(\mu, \nu) = \sup_{f: X \rightarrow \mathbf{R}} \mathbb{E}_{X \sim \mu} f(X) - \mathbb{E}_{X \sim \nu} f(X),$$

$$\|f\|_{\text{Lip}} \leq 1$$

define the signature-Wasserstein distance between (distributions on) time series

$$\text{Sig-}W_1(\mu, \nu) = \sup_{\substack{L \text{ linear} \\ \|L\|_2 \leq 1}} L \left(\mathbb{E}_{X \sim \mu} S(X) - \mathbb{E}_{X \sim \nu} S(X) \right)$$

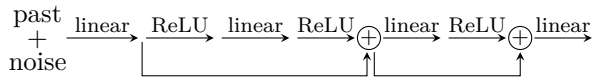
$$= \left\| \mathbb{E}_{X \sim \mu} S(X) - \mathbb{E}_{X \sim \nu} S(X) \right\|_2$$

$$\approx \left\| \mathbb{E}_{X \sim \mu} S_M(X) - \mathbb{E}_{X \sim \nu} S_M(X) \right\|_2$$

(this is the MMD (maximum mean discrepancy) with the truncated signature S_M as features).

For time series continuation, $\mathbb{E}[S(X_{>t})|X_{\leq t}]$ can be computed by linear regression $S(X_{>t}) \sim S(X_{\leq t})$.

Use as a discriminator, with generator:



To compute the signature: `signatory` (Python).

**Fractional SDE-Net: generation
of time series data with long-term memory**
K. Hayashi and K. Nakagawa (2022)

Neural ODEs can be generalized to neural SDEs (by adding noise) and neural fractional SDEs (if $H \neq \frac{1}{2}$, Itô calculus no longer applies but, for $H > \frac{1}{2}$, we can still define a stochastic integral).

**Holdout-based fidelity and privacy assessment
of mixed-type synthetic data**
M. Platzer and T. Reutterer

To measure fidelity and privacy risk of generated data, look at:

- 1- and 2-dimensional distributions;
- The distance to the closest observation in the training data

and compare them with the holdout set.

Multi-asset spot and option market simulation
M. Wiese et al.

Generate synthetic time series with normalizing flows

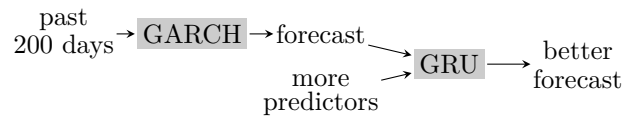
$$(\mathcal{F}_t, Z_{t+1}) \mapsto X_{t+1}$$

$$Z_{t+1} \leftarrow (\mathcal{F}_t, X_{t+1})$$

where X is observed, Z latent, and $\mathcal{F}_t = (X_t, X_{t-1}, \dots)$. For option prices, enforcing the no-arbitrage condition complicates things.

**A hybrid deep learning approach to
purchasing strategy of carbon emission rights**
J. Xu and Y. Fu

Stack GARCH and GRU to forecast carbon emission prices.



Predictors include carbon markets, equity index, FX, energy (oil, etc.), futures, industrial indices, air quality.

**Cryptocurrency valuation:
an explainable AI approach**
Y. Liu and L. Zhang

“Fundamental ratios” for crypto currencies, such as miners’ earnings/price, market value/traded value, market value/number of users², are bad predictors of future returns. Instead, the utility/price ratio has some predictive power on long-term returns.

$$\text{utility} = \frac{\text{velocity} \times \text{staking ratio}}{\text{volatility} \times \text{dilution rate}}$$

$$\text{velocity} = \frac{\text{tokens traded (past day)}}{\text{tokens available}}$$

$$\text{staking ratio} = \frac{\text{inactive tokens (past year)}}{\text{tokens available}}$$

$$\text{dilution rate} = \text{token supply annual growth rate}$$

**Price impact of order flow imbalance:
multi-level, cross-sectional and forecasting**
R. Cont et al. (2021)

The order flow imbalance (OFI) is the difference between:

- The change in bid volume at level m and
- The change in ask volume at level m ;

normalize by dividing with the volume in levels up to m ; aggregate by taking the first principal component of $\text{ofi}_1, \dots, \text{ofi}_{10}$; use to explain contemporaneous returns, or forecast future returns; also try to use the ofi_m or $\text{PC}_1(\text{ofi})$ of other stocks (lasso).

Analysis of a five-factor capital market model
S.F. Jarnier and M. Preisel (2017)

The Munk capital market model is

$$\begin{aligned} dr &= \kappa(\bar{r} - r)dt + \sigma dW && \text{short interest rate} \\ dS/S &= (r + x)dt + \sigma dW && \text{stock index} \\ dx &= \alpha(\bar{x} - x)dt + \sigma dW && \text{equity risk premium} \\ dI/I &= \pi dt + \sigma dW && \text{price index (inflation)} \\ d\pi &= \beta(\bar{\pi} - \pi)dt + \sigma dW && \text{expected inflation} \end{aligned}$$

Optimal portfolio choice and stock centrality for tail risk events
C. Katsouris (2021)

Replace the variance matrix with the VaR matrix:

$$\begin{aligned} P[R_i \leq \text{VaR}_i^\alpha] &= \alpha \\ P[R_i \leq \text{CoVaR}_{ij}^\alpha | R_j \leq \text{VaR}_j^\alpha] &= \alpha \\ \Delta \text{CoVaR}_{ij} &= \text{CoVaR}_{ij} - \text{VaR}_{ij} \\ V_{ii} &= \text{VaR}_i^+ \\ V_{ij} &= \sqrt{\text{VaR}_i^+ \text{VaR}_j^+ \Delta \text{CoVaR}_{ij}} \\ \tilde{V} &= \frac{1}{2}(V + V') \end{aligned}$$

Optimal trend following portfolios
S. Valeyre (2022)

To estimate the trend, also use inter-asset cross-correlations. The maximum Sharpe portfolio is then $w \approx C^{-1}(\alpha C_\xi + \beta \mu \mu') C^{-1}$, where C is the variance matrix of the asset returns, and C_ξ that of the stochastic trend.

$$\begin{aligned} X_{it} &= \mu_i + \sum_{s < t} S_{t-s} \xi_{is} + \varepsilon_{it} \\ C_t &= \text{Var } X_{\cdot, t} \quad C_{\xi t} = \text{Var } \xi_{\cdot, t} \end{aligned}$$

The agnostic risk parity portfolio is the risk parity portfolio on the first eigenvectors of the variance matrix.

From tether to libra: stablecoins, digital currency and the future of money
A. Lipton et al.

Stable coins are often classified according to their collateral (fiat-, commodity-, crypto- or uncollateralized). Instead, look if the ability to redeem the coin is guaranteed by a legal framework, technology, or not guaranteed.

Combining reinforcement learning and inverse reinforcement learning for asset allocation recommendations
I. Halperin et al.

IRL learns a reward function from trajectories. T-REX is an IRL algorithm learning the reward function from non-optimal trajectories: for this, it relies on extra information, viz a ranking of all trajectories.

Combine RL and IRL:

- Learn the reward function of a group of portfolio managers (following the same type of strategy);
- Solve the corresponding RL problem, to improve their decisions.

To deal with the paucity of data, aggressively reduce the dimension to sector weights (for the benchmark and the strategies).

Limiting spectral distribution of high-dimensional Hayashi-Yoshida estimator of integrated covariance matrix
A. Chakrabarti and R. Sen (2022)

The Hayashi-Yoshida estimator of the integrated covariance matrix

$$\Sigma_{HY}^{X,Y} = \sum_{ij} \Delta X_i \cdot \Delta Y_j \cdot \mathbf{1}_{I_i^X \cap I_j^Y \neq \emptyset}$$

is inconsistent and unreliable in high dimension.

Tone at the top? Quantifying management presentation
Wolfe Research (2018)

Extract the following features from call transcripts:

- Readability
- Tone (Harvard-IV, Loughran-McDonald, Vader)
- POS (proportion of adjectives, adverbs (subjectivity), first-person singular vs plural pronouns)
- Ratio of digits to letters
- Number of analysts participating
- Topics (LDA), after removing sector-specific words or phrases (otherwise, the topics just identify the sectors)
- Change in those features.

and forecast future returns or earnings surprises.

Text mining form 8-K disclosures
Wolfe Research (2020)

Companies need to report “material information” within 4 days in SEC form 8-K. Forecast future returns (event study) with:

- Filing frequency;
- Positive and negative sentiment;
- Readability;
- Proportion of numbers in the text;
- Filing delay;
- Event abnormal return;
- Event type (SEC classification).

Network of economically-linked firms
Wolfe Research (2021)

Use the average momentum (k -month, or k -month minus 1-month) of nearby firms to forecast future returns, where “nearby” is determined by:

- Business segment

- Geographic exposure (US, CA, EU, UK, ANZ, AxJ, JP, CN, EMEA, LATAM, rest);
- Number of sell-side analysts in common;
- Key executives
- Patent citations;
- Supply chain (bipartite graph of landlords and tenants for REITs in the Russell 3000);
- Shipping network.

***NLP 5G Evolution:
text mining global corporate filings
Wolfe Research (2019)***

Compute the following features on 10K/10Q filings:

- Conformity to the SEC's requirements;
- YoY or QoQ similarity with the previous one (cosine similarity on bags of words, [try (sliced) Wasserstein distance]);
- POS (adjectives, adverbs, pronouns).

Forecast future returns with an ensemble of elastic net and xgboost.

For international companies, build a multilingual sentiment lexicon.

For languages with no spaces: `jieba` (Chinese), `tinysegmenter` (Japanese), `konlpy` (Korean).

***Patent, innovation and alpha
Wolfe Research (2019)***

Innovative industries (more than 50% of the companies have patents) outperform.

Define the R&D intensity as $R\&D/Sales$, or $\log(R\&D/MCap)$, or the residuals of $\log(R\&D) \sim \log(MCap)$.

Compute features from patents:

- Innovation intensity: number of patents in a 3-year window;
- Originality: (effective) number of patent classes or subclasses;
- Number of citations, back-citations, number of unique assignees in citations or back-citations, divided by the number of patents;
- Number of foreign applications;
- Maintenance fee non-payment;
- HITS, PageRank.

Adjust those factors for size (regress against $\log(MCap)$).

***Optimal expected utility risk measures
S. Geissel et al. (2017)***

Risk measures defined from a utility function include:

$$\begin{aligned}\rho^{SR}(X) &= \inf\{\eta : E[u(X + \eta)] \geq q\} \\ \rho^{CE}(X) &= -CE[X] = -u^{-1} E[u(X)] \\ \rho^{OCE}(X) &= -\sup_{\eta} -\eta + E[u(X + \eta)] \\ \rho^{OEU}(X) &= -\sup_{\eta} -\eta + \alpha CE[X + \eta] \quad \alpha \in (0, 1)\end{aligned}$$

***Deep quantile and
deep composite model regression
T. Fissler et al. (2021)***

Quantile regression, for several quantiles, can be fitted with a neural network, with constraints to ensure the quantiles are in the correct order. This can be generalized to “ES regression”: the expected shortfall (ES) is not elicitable (not an M-estimator), but the triplet (quantile, lower ES, upper UE) is.

$$\left(F^{-1}(\tau), \frac{1}{\tau} \int_0^{\tau} F^{-1}, \frac{1}{1-\tau} \int_{\tau}^1 F^{-1}\right)$$

***2T-POT Hawkes model
for dynamic left- and right-tail
quantile forecasts of financial returns
M.F. Tomlinson et al. (2022)***

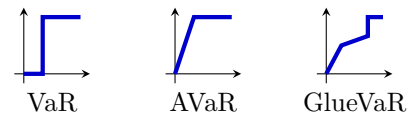
Model quantiles of a time series with a (2-tailed) peaks-over-threshold Hawkes model (Hawkes model with generalized Pareto excess magnitudes – the GPD parameters are constant) rather than a GARCH model.

***Multinomial backtesting
of distortion risk measures
S. Bettels et al. (2022)***

The *distorsion risk measure* defined by a decreasing $g : [0, 1] \rightarrow [0, 1]$ is

$$\begin{aligned}\rho_g(X) &= \int_{-\infty}^0 (g(P[X > x]) - 1)dx + \int_0^{\infty} g(P[X > x])dx \\ &= \int_0^1 q_X^+(1-u)dg(u) \\ &= \int_0^1 F_X^{-1}(1-u)dg(u) \\ q_X^+(u) &= \sup\{x : F_X(u) \leq u\}.\end{aligned}$$

Examples include:



$$\begin{aligned}g(u) &= 1 - (1 - u)^n & \text{MinVaR: } E[\text{Max}(X_1, \dots, X_n)] \\ g(u) &= u^{1/n} & \text{MaxVaR} \\ g(u) &= 1 - (1 - u^{1/n})^n & \text{MinMaxVaR} \\ g(u) &= (1 - (1 - u)^n)^{1/n} & \text{MaxMinVaR} \\ & \text{etc.}\end{aligned}$$

**Decomposable sums and their implications
on naturally quasiconvex risk measures**
Ç. Ararat et al. (2022)

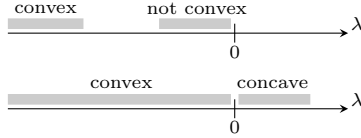
A risk measure is *naturally convex* if

$$\forall X, Y \quad \forall \lambda \quad \exists \mu \quad \rho(\lambda X + (1-\lambda)Y) \leq \mu \rho(X) + (1-\mu)\rho(Y).$$

The *convexity index* of $f : \mathbf{R}^m \rightarrow \bar{\mathbf{R}}$ is

$$c(f) = \sup\{\lambda < 0 : r_\lambda \text{ convex}\} \\ \text{or } c(f) = \sup\{\lambda \geq 0 : r_\lambda \text{ concave}\}$$

where $r_\lambda = e^{-\lambda f(x)}$.



A function of the form

$$s(x_1, \dots, x_n) = f_1(x_1) + \dots + f_n(x_n)$$

is quasi-convex iff

$$c(f_1) + \dots + c(f_n) \geq 0,$$

iff either all the f_i 's are convex, or at most one is non-convex and

$$\frac{1}{c(f_1)} + \dots + \frac{1}{c(f_n)} \leq 0.$$

If the f_i 's are convex,

$$\frac{1}{c(s)} = \sum \frac{1}{c(f_i)}.$$

**Model aggregation for risk evaluation
and robust optimization**
T. Mao et al. (2022)

Given a risk measure $\rho : \mathcal{M} \rightarrow \mathbf{R}$ and an uncertainty set $\mathcal{F} \subset \mathcal{M}$, the worst case risk is

$$\rho^{\text{WR}} = \sup_{F \in \mathcal{F}} \rho(F).$$

Instead, given an order relation \preceq on \mathcal{M} , e.g., first or second order dominance,

$$F \preceq_1 G \text{ iff } \forall u \text{ increasing } \int u dF \leq \int u dG$$

$$F \preceq_2 G \text{ iff } \forall u \text{ increasing convex } \int u dF \leq \int u dG$$

one can consider the *model aggregation risk*

$$\rho^{\text{MA}}(\mathcal{F}) = \rho\left(\bigvee_{F \in \mathcal{F}} F\right).$$

For \preceq_1 and \preceq_2 , the supremum is easy to compute:

$$\begin{aligned} \bigvee_1 \mathcal{F} &= \inf_{F \in \mathcal{F}} F && \text{cdf} \\ \left(\bigvee_1 \mathcal{F}\right)^{-1} &= \sup_{F \in \mathcal{F}} F^{-1} && \text{icdf} \\ \pi_{\bigvee_2 \mathcal{F}} &= \sup_{F \in \mathcal{F}} \pi_F && \text{integrated survival function} \\ \bigvee_2 \mathcal{F} &= 1 + \left(\sup_{F \in \mathcal{F}} \pi_F\right)'_+ \end{aligned}$$

where the integrated survival function is

$$\pi_F = \mathbb{E}_{X \sim F}[(X - x)_+] = \int_x^\infty (1 - F(t)) dt.$$

Under reasonable assumptions on \mathcal{F} ,

$$\begin{aligned} \text{VaR}_\alpha^{\text{MA}, \preceq_1} &= \text{VaR}_\alpha^{\text{WR}} \\ \text{ES}_\alpha^{\text{MA}, \preceq_2} &= \text{ES}_\alpha^{\text{WR}}. \end{aligned}$$

Other risk measures include

$$\text{RVaR}_{\alpha, \beta}(F) = \frac{1}{\alpha - \beta} \int_\alpha^\beta \text{VaR}_s(F) ds \quad \text{range VaR}$$

$$\text{PD}_k(F) = \int_0^1 k s^{k-1} \text{VaR}_s(F) ds \quad \text{power distortion}$$

or the *expectile*, defined as the unique solution $t = \text{ex}_\alpha(F)$ of

$$\alpha \mathbb{E}_{X \sim F}[(X - t)_+] = (1 - \alpha) \mathbb{E}_{X \sim F}[(X - t)_-].$$

Common uncertainty sets include

– The Wasserstein ball around F_0 :

$$W_p(F, F_0) = \left(\int_0^1 |F^{-1}(s) - F_0^{-1}(s)|^p ds \right)^{1/p} \quad \text{dim}=1$$

$$W_p(F, F_0) = \inf_{\substack{X \sim F \\ Y \sim F_0}} (\mathbb{E} \|X - Y\|_a^p)^{1/p}$$

– The mean-variance uncertainty set

$$\mathcal{F}_{\mu, \sigma} = \{F \in \mathcal{M} : \mathbb{E}[F] = \mu \text{ and } \text{Var}[F] = \sigma^2\}$$

**Influence functions
for risk and performance estimators**
S. Zhang et al. (2020)

The functional of an estimator $\hat{\theta}$ is

$$\theta(F) = \lim_{\substack{n \rightarrow \infty \\ r_i \sim F}} \hat{\theta}_n(r_1, \dots, r_n).$$

Its influence function is

$$\begin{aligned} \text{IF}(r, \theta, F) &= \left. \frac{d}{d\gamma} \theta(F_\gamma) \right|_{\gamma=0} \\ F_\gamma &= (1 - \gamma)F + \gamma \delta_r \end{aligned}$$

It satisfies

$$\mathbb{E}_{r \sim F} [\text{IF}(r, \theta, F)] = 0$$

$$\hat{\theta}_n - \theta(F) = \frac{1}{n} \sum \text{IF}(r_i, \theta, F) + \text{remainder}$$

hence

$$\text{Var}[\hat{\theta}_n] \approx \frac{1}{n} \sum \text{IF}^2(r_i, \theta, F_n)$$

(the variance of the estimator is the variance if the IF-transformed returns).

We can compute the IF of most performance measures (Sharpe, Sortino, Rachev, Omega ratios, etc.) and therefore estimate their standard deviation (assuming iid returns).

R implementation in RPEIF and RPESE.

Standard errors of risk and performance estimators for serially correlated returns
X. Chen and R.D. Martin (2019)

If the returns are not iid, the variance should be corrected with the ACF,

$$\text{Var}[\hat{\theta}_n] = \text{E}[\text{IF}^2(r)] + 2 \sum_{k \geq 1} \text{E}[\text{IF}(r_0)\text{IF}(r_k)]$$

which can be computed from the periodogram, which can be smoothed with a GLM model for exponential distributions (polynomial regression with a lasso penalty to control the degree).

PerformanceAnalytics: estimation of higher order moments
D. Cornilly and B.G. Peterson (2020)

Factor model for the skewness and kurtosis tensors.

Dependence model assessment and selection with DecoupleNets
M. Hofert et al. (2022)

Learn a transformation from a known copula in dimension $d \geq 2$ to the uniform copula in dimension 2, and use it for gof tests (try several copulas, and keep that for which the result is closest to uniform).

Continuous-time stochastic gradient descent for optimizing over the stationary distribution of stochastic differential equations
Z. Wang and J. Sirignano (2022)

To find the parameters θ of an SDE

$$dX = \mu_\theta(X)dt + \sigma_\theta(X)dW$$

to minimize

$$F(\theta) = \left\| \mathbb{E}_{Y \sim \pi_\theta} [f(Y) - \beta] \right\|^2$$

where $E[f(Y)]$ are some generalized moments of the invariant distribution π_θ :

$$\begin{aligned} \partial_t \theta &= -2\alpha [f(\bar{X}) - \beta] \tilde{X}' \nabla_x f(X) \\ d\tilde{X} &= (\tilde{X} \partial_x \mu + \partial_\theta \mu)dt + (\tilde{X} \partial_x \sigma + \partial_\theta \sigma)dW \\ dX &= \mu dt + \sigma dW \\ d\bar{X} &= \mu dt + \sigma d\bar{W} \end{aligned}$$

(\tilde{X} estimates $\partial_\theta X$, α is the learning rate).

Multivariate matrix-exponential affine mixtures and their applications in risk theory
E.C.K. Cheung et al. (2021)

The matrix-exponential distribution on $[0, \infty)$, $X \sim \text{ME}(\alpha, T, t)$ has density $f(x) = \alpha e^{T^x t}$, $\alpha \in \mathbf{R}^{1 \times p}$, $T \in \mathbf{R}^{p \times p}$, $t \in \mathbf{R}^{p \times 1}$ and cdf $F(x) = 1 + \alpha e^{T^x t} T^{-1} t$. These are the distributions whose Laplace transform is

rational. They are stable by mixtures and affine mixtures (mixtures $f = \sum c_i f_i$, with potentially negative weights c_i as long as $f \geq 0$).

Multivariate matrix-exponential affine mixtures have density

$$f(x_1, \dots, x_M) = \sum_{1 \leq i_1, \dots, i_M \leq L} p_{i_1, \dots, i_M} f_{i_1}(x_1) \cdots f_{i_M}(x_M).$$

Graph neural networks for asset management
G. Pacreau et al.

Feed pricing data (and your model's alpha) to an LSTM, and the result to a GCN or GAT (or the hypergraph equivalent) to forecast future returns (or their signs), with a penalty to get the correct order (learning to rank)

$$\lambda \sum_{ij} [(\hat{y}_i - \hat{y}_j)(y_i - y_j)]_+.$$

Portfolio selection models based on interval-valued conditional value at risk (ICVaR) and empirical analysis
J. Zhang and K. Zhang (2022)

To extend portfolio construction to return intervals $[x^L, x^U]$, use the elxicographic order of (mean, left) for the objective and replace the constraint $Ax \leq B$ with $a^U x \leq b^U$, $\Gamma(B, Ax) \leq \gamma$, where

$$\begin{aligned} \Gamma(A, B) &= \frac{m(B) - m(A)}{w(B) + w(A)} \\ A &= [a^L, a^U] \\ B &= [b^L, b^U] \\ m(A) &= \frac{1}{2}(a^L + a^U) \\ w(A) &= a^U - a^L \\ \gamma &\in [0, 1] \text{ fixed.} \end{aligned}$$

VaR and CVaR can be similarly generalized to interval-valued random variables.

The growth of relative wealth and the Kelly criterion
A.W. Lo et al. (2017)

The Kelly portfolio f , for two assets a and b with ratio returns X_a and X_b , maximizes

$$\mu(f) = \text{E}[\log(1 + fX_a + (1 - f)X_b)]$$

and satisfies

$$\text{E} \left[\frac{X_a - X_b}{1 + fX_a + (1 - f)X_b} \right] = 0$$

(unless $\text{E}[(1 + X_b)/(1 + X_a)] < 1$ or $\text{E}[(1 + X_a)/(1 + X_b)] < 1$, in which case $f = 1$ or $f = 0$).

Alternatively, given two investors f and g , we can look for f maximizing the relative final wealth $W_f/(W_f + W_g)$.

Fat tails and optimal liability driven portfolios

J. Rosenzweig

In portfolio optimization, replace the variance with some other even moment (larger exponents give an approximation of the minimum return).

Traversing the local polytopes of ReLU neural networks: a unified approach for network verification

S. Xu et al.

A ReLU network defines a piecewise linear function. Each neuron splits the space along a hyperplane: this partitions the space into polytopes, each defined by a set of inequalities; non-redundant inequalities (flipping them gives a non-empty polytope) define the faces of that polytope. Neighbouring polytopes (they differ by the direction of one non-redundant inequality) form a *polytope graph*, which can be explored with breadth-first search (BFS). This graph, with BFS traversal, can be used to prove properties of the network, e.g., existence of local adversarial attacks, or to generate counterfactual samples. Previous approaches used mixed integer programming (MIP), but the polytope graph can be explored with linear programming (LP) alone (to check if a set of linear inequalities defines the empty set).

Explainable neural networks based on additive index models

J. Vaughan et al. (2018)

The explainable neural network (xNN) is an *additive index model*

$$f(x) = \mu + \sum \gamma_i h_i(\beta_i x)$$

where the h_i 's are scalar functions defined by neural nets.

Enhancing explainability of neural networks through architecture constraints

Z. Yang et al.

The generalized additive index model (GAIM, aka projection pursuit learning)

$$f(E[y|x]) = \mu + \sum h_i(w'_i x)$$

can be estimated with neural networks h_i ,

$$f(E[y|x]) = \mu + \sum \beta_i h_i(w'_i x),$$

with an L^1 penalty on both β and w , a smooth penalty on h_i , $\text{Mean}_x h_i''(w_i x)^2$, a normalization constraint for the h_i 's,

$$E[h_i(z)] = 0, \quad E[h_i(z)^2] = 1,$$

imposed by a batchnorm layer, and an orthogonality constraint $W'W = I_k$, i.e., $W \in \text{Stiefel}(p, k)$, imposed with the *Cayley transform*, i.e., by replacing the gradient step

$$W \leftarrow W - \tau G, \quad G = \nabla_W \mathcal{L}$$

with

$$A = GW' - WG'$$

$$W \leftarrow \left(I + \frac{\tau}{2} A \right)^{-1} \left(I - \frac{\tau}{2} A \right) W.$$

GAMI-Net: an explainable neural network based on generalized additive models with structured interactions

Z. Yang et al.

One can estimate a GAM, with neural networks instead of (1- and 2-dimensional) splines, and with:

- Pruning of less important main effects (proportion of explained variance);
- Pairwise interactions, estimated on the residuals of the main effects model, only for pairs for which one variable has already been selected, chosen by looking at the proportion of variance explained by forests of shallow trees on each pair, pruned as above;
- An orthogonality penalty

$$\sum_{ij} \text{Mean } f(x_i) f(x_i, x_j)$$

to clearly separate the main effects from the interactions.

Unwrapping the black box of deep ReLU networks: interpretability, diagnostics and simplification

A. Sudjianto et al.

A neural network with ReLU activations is a locally linear model:

- Look at the parallel plot of the coefficients of those linear functions;
- For each feature x_i , plot the segments $\beta_i x_i \sim x_i$, one for each cell (“activation region”).

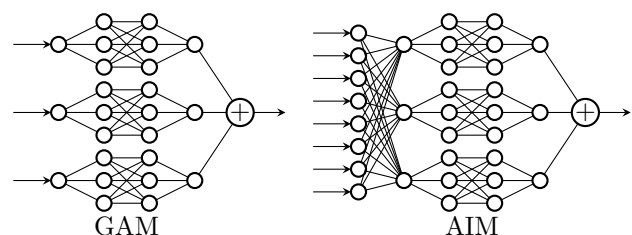
To simplify the model, merge nearby cells with similar coefficients (not unlike the fused lasso).

Python implementation: `aletheia-dnn`.

Adaptive explainable neural networks (AxNNs)

J. Chen et al. (2020)

Fit an AIM (additive index model) on the residuals of a GAM (generalized additive model), both estimated with neural networks: the GAM estimates the main effects, the AIM the residual interactions. Train with AdaBoost, with 1-variable GAMs and 2-variable AIMs as weak learners.



***Surrogate locally-interpretable models
with supervised machine learning algorithms***
L. Hu et al. (2020)

A regression tree (with linear models with an L^1 penalty, or GAMs, in the leaves), trained on the output of a neural net, gives an interpretable surrogate model, not unlike LIME (a linear model on a neighbourhood of each observation) or KLIME (a linear model on each k -means cluster – k -means is an unsupervised clustering algorithm, while a regression tree is a supervised one).

***Designing inherently
interpretable machine learning models***
A. Sudjianto and A. Zhang (2021)

To measure the interpretability of a model, look

- if constraints such as additivity, sparsity, linearity, smoothness, monotonicity can be imposed, globally or locally, at least approximately,
- if the model can be visualized,
- if it uses an (orthogonal) projection of the input features,
- if it segments the input space in a small number of regions.

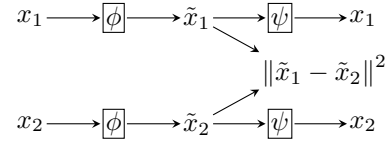
***General pitfalls
of model-agnostic interpretation methods
for machine learning models***
C. Molnar et al.

- Confusion between feature effect (Shapley score) and feature importance (e.g., permutation feature importance, PFI): use both;
- Interpreting a model that does not generalize well: this gives an interpretation of the model, not of the data generation process;
- Non-intrinsically interpretable models – try GAM boosting;
- Ignoring feature dependence: most methods assume features are independent;
- Using only correlation to measure dependence: check KCCA (kernel canonical correlation), HSIC (Hilbert-Schmidt independence criterion), MI (mutual information – more difficult to estimate);
- Ignoring interactions (when aggregated, significant effects can cancel out): look at ICE curves;
- Ignoring uncertainty: use the bootstrap;
- Drawing causal conclusions.

Learning Wasserstein embeddings
N. Courty et al. (2018)

A *Deep Wasserstein embedding* is a mapping (of probability distributions, or images – the examples given are MNIST and doodles) transforming the Wasserstein distance into the Euclidean distance, trained on data ($x_i = (x_i^1, x_i^2)$, $y_i = d_W(x_i^1, x_i^2)$), with a Siamese network, regularized with a decoder and a reconstruction loss.

Applications include Wasserstein barycenter (interpolation) and principal geodesic analysis.



***Set representation learning
with generalized sliced-Wasserstein embeddings***
N. Naderializadeh et al.

To process sets with a neural net, one usually uses permutation equivariant modules (e.g., self-attention, or, more generally, processing each point in the same way) and permutation invariance modules (multi-head attention, pooling): DeepSets, SetTransformers. Instead:

- View the elements of a set as samples from a probability distribution μ_i ;
- Project them on random 1-dimensional subspaces, $\nu_{i\theta} = g_\theta \# \mu_i$;
- Compute the Monge map between each $\nu_{i\theta}$ and some reference distribution $\nu_{0\theta}$: $F_{\nu_{i\theta}}^{-1} \circ F_{\nu_{0\theta}}$;
- Use the cumulative distribution transforms $F_{\nu_{i\theta}}^{-1} \circ F_{\nu_{0\theta}} - \text{Id}$ as representation of the μ_i .

The projection g_θ , can be learned from data, using contrastive learning (SimCLR, SiamSiam).

Wasserstein embedding for graph learning
S. Kolouri et al. (2021)

Represent a graph with its node embedding, *i.e.*, a set of points in \mathbf{R}^d , and then, the linear Wasserstein embedding.

Do not compute all pairwise distances, but only distances to some reference graph.

***Generalized sliced distances
for probability distributions***
S. Kolouri et al.

The *Radon transform* recovers a distribution p from its slices p_f along hyperplanes

$$p_{f_\theta}(t) = \int_{\mathbf{R}^d} p(x) \delta(t - \langle x, \theta \rangle) dx$$

$$p(x) = \int_{\mathbf{S}^{d-1}} (p_{f_\theta} * \eta)(\langle x, \theta \rangle) d\theta$$

where $\hat{\eta}(\omega) = c|\omega|^{d-1}$.

The *generalized Radon transform* uses a family of hypersurfaces $H = \{x \in \mathbf{R}^d : f_\theta(x) = t\}$ instead, where the f_θ are called “defining functions”. A metric ξ between 1-dimensional probability measures can be extended to higher dimensions

$$\zeta(p, q) = \left(\int \xi(p_{f_\theta}, q_{f_\theta})^r d\theta \right)^{1/r}, \quad r \geq 1$$

(generalized sliced probability metric) or

$$\zeta^*(p, q) = \sup_{\theta} \xi(p_{f_{\theta}}, q_{f_{\theta}}),$$

For instance:

$$\begin{aligned}\zeta^2(p, q) &= \int \|Ap_{f_{\theta}} - Aq_{f_{\theta}}\| d\theta \\ A &= \text{Id} \\ Ap(t) &= \int_{-\infty}^t p(s) ds \\ Ap(t) &= \int_{-\infty}^{\infty} p(t)k(t, s) ds.\end{aligned}$$

**Parameter prediction
for unseen deep architecture
B. Knyazev et al. (2021)**

Hypernetworks learn to compute the parameters of another neural network from training data in a single forward pass:

training data \mapsto weights

but they require the neural net to have a fixed architecture. Graph neural nets (GNN) allow an arbitrary architecture (for a fixed set of node types):

training data, architecture \mapsto weights.

**Squeeze-and-excitation networks
J. Hu et al.**

A squeeze-and-excite block combines:

- A convolution layer, outputting C channels;
- Global averaging, separately for each channel, outputting a vector of length C ;
- A simple transformation,

$$z \mapsto s = \sigma(w_2 \text{ReLU}(w_1 z)),$$

to compute the channel-specific scale, outputting a vector of length C ;

- Multiplication of the convolution output and the scale.

**Learning in high dimension
always amounts to extrapolation
R. Balestriero**

Interpolation occurs when a new sample is in the convex hull of the training data; in high dimension, this is unlikely to happen.

$$\begin{aligned}x_1, \dots, x_N &\sim \text{Unif}(B_d) \\ x &\sim \text{Unif}(B_d) \\ P[x \in \text{chull}(x_1, \dots, x_N)] &\xrightarrow[N < 2^{d/2}/d]{d, N \rightarrow \infty} 0\end{aligned}$$

**Task-based end-to-end model learning
in stochastic optimization
P.L. Donti et al.**

Traditional machine learning, for

x : features $f(x, y, z)$: loss
 y : predictions
 z : policy

is often a 2-step process

$$\begin{aligned}\theta &= \underset{\theta}{\text{Argmin}} \mathbb{E}_{x, y \sim \text{data}} [-\log p(y|x; \theta)] \\ z(x) &= \underset{x}{\text{Argmin}} \mathbb{E}_{y \sim p(y|x; \theta)} [f(x, y, z)].\end{aligned}$$

Black-box, end-to-end optimization is not very data-efficient:

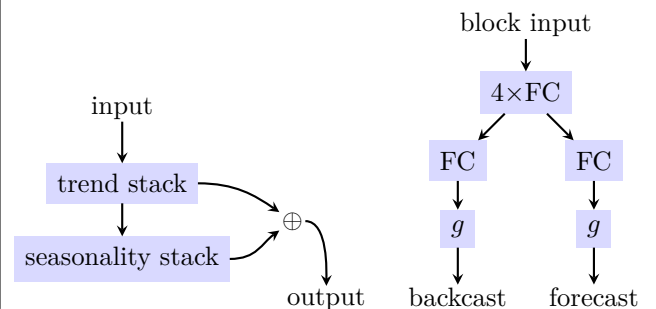
$$\underset{\theta}{\text{Minimize}} \mathbb{E}_{x, y \sim \text{Data}} [f(x, y, z(x; \theta))].$$

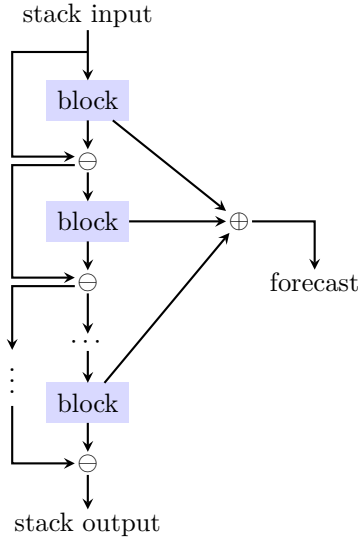
Modeling the data distribution (instead of the policy) in an end-to-end fashion (*i.e.*, with the correct loss function, instead of the log-likelihood) is more data-efficient. Iterate:

$$\begin{aligned}x, y &\sim \text{Data} \\ z^*(x, \theta) &= \underset{z}{\text{Argmin}} \mathbb{E}_{y \sim p(y|x; \theta)} f(x, y, z) \\ \theta &\leftarrow \theta - \alpha \nabla_{\theta} f(x, y, z^*(x, \theta)).\end{aligned}$$

**N-Beats: neural basis expansion analysis
for interpretable time series forecasting
B.N. Oreshkin et al. (2020)**

To forecast time series, stack blocks with two outputs, to forecast/backcast the next/previous periods, taking the residual of the previous block as input; the nonlinearities are given by basis function expansions $\sum \theta_i \mathbf{v}_i$, to extract the trend (first stack) and the seasonality (second stack).





Ensemble several models: different loss functions (SMAPE, MASE, MAPE), window lengths, random initialization.

How powerful are graph neural networks?

K. Xu et al. (2019)

For $\varepsilon \in \mathbf{R} \setminus \mathbf{Q}$, there exists $f : \mathbf{R} \rightarrow \mathbf{R}$ such that

$$\begin{cases} \mathbf{Q} \times \text{Multisets}(\mathbf{Q}) & \rightarrow \mathbf{R} \\ (u, (v_i)_i) & \mapsto (1 + \varepsilon)f(u) + \sum_i f(v_i) \end{cases}$$

be injective. This suggests the following aggregation, for graph neural nets (GIN):

$$h_v \leftarrow \text{MLP}\left((1 + \varepsilon)h_v + \sum_{u \in N(v)} h_u\right)$$

where ε is learned.

Principal neighbourhood aggregation for graph nets

G. Corso et al. (2020)

Do not use a single aggregation operator, but several (mean, standard deviation, minimum, maximum), and rescale them with the degree (on a logarithmic scale, otherwise the messages would be amplified or attenuated exponentially).

$$\left(\frac{\log(d+1)}{\langle \log(d+1) \rangle} \right)^{+1, -1, \text{ or } 0}$$

Learning aggregation functions

G. Pellegrini et al. (2021)

Principal neighbourhood aggregation (PNA) uses a finite set of ($4 \times 3 = 12$) aggregations. Instead, try

$$\text{LAF}(x) = \frac{\alpha L_{a,b}(x) + \beta L_{c,d}(1-x)}{\gamma L_{d,f}(x) + \delta L_{g,h}(1-x)}$$

$$L_{a,b}(x) = \left(\sum_i x_i^b \right)^a \quad x \in [0, 1]^n$$

Neural message passing for quantum chemistry

J. Gilmer et al. (2017)

$$m_v \leftarrow \sum_{w \in N(v)} M(h_v, h_w, e_{vw})$$

$$h_v \leftarrow U(h_v, m_v)$$

Rethinking graph transformers with spectral attention

D. Kreuzer et al. (2021)

Transformers use positional embeddings, with sines and cosines. For graphs, we could use the first eigenvectors of the Laplacian (GT), but this ignores the eigenvalues. Instead, SAN learns the positional embedding: for each node, transform the eigenvalues $\lambda_1, \dots, \lambda_m$ and eigenvectors $\phi_{1j}, \dots, \phi_{mj}$ as

$$(2 \times m) \xrightarrow[2 \times k]{\text{linear}} (k \times m) \xrightarrow[m \times m]{\text{transformer}} (k \times m) \xrightarrow{\text{sum}} (k \times 1).$$

Randomly flip the sign of the eigenvectors to make the model sign-invariant.

A generalization of transformer networks to graphs

V. P. Dwivedi and X. Bresson (2021)

The graph transformer (GT) layer uses the low-frequency eigenvectors of the Laplacian as positional encoding.

Training graph neural networks with 1000 layers

G. Li et al. (2021)

Use reversible (block-triangular) connections.

A dual-stage attention-based recurrent neural network for time series prediction

Y. Qin et al.

Stack two LSTM layers with attention (DARNN).

$$\alpha_{st} \propto f(h_{t-1}, x_t) \quad s < t$$

$$h_t = \text{LSTM}(h_{t-1}, \sum_{s < t} \alpha_{st} x_s)$$

$$\beta_{st} \propto g(\ell_{t-1}, h_s) \quad s < t$$

$$d_t = \text{LSTM}(d_{t-1}; y_{t-1}, \sum_{s < t} \beta_{st} h_s)$$

$$\hat{y}_T = \phi(d_T, \sum_{s < T} \beta_{sT} h_s)$$

Gradients are not all you need

L. Metz et al. (2021)

Exact gradients (from automatic differentiation) may be less informative than approximate (black-box) ones.

This phenomenon appears, in particular, in iterated systems: RNNs, rigid body physics, ODE solvers, optimization layers, etc.

To limit the problem: initialize the RNNs close to the identity, use LSTMs, use conservation laws (and other known properties), try truncated BPTT, truncated gradients, or, even black-box gradients.

Top2vec: distributed representations of topics

D. Angelov

top2vec = doc2vec + UMAP + HDBScan

Supervised linear dimension reduction

methods: review, extensions and comparisons

S. Xu et al.

Prefer PLS (an eigen decomposition) or LSPCA (manifold optimization).

PCA: Minimize $\|X - XU U'\|^2$
s.t. $U'U = I$

PLS: Maximize $[(Xu)'y]^2$
s.t. $\|u\|=1$
(and iterate after projecting X and y on u^\perp)

LSPCA: Minimize $\|y - XU\beta\|^2 + \lambda \|X - XU U'\|^2$
s.t. $U'U = I$

A new coefficient of correlation

S. Chatterjee

To assess if $Y = f(X)$ (for f measurable but not necessarily monotonic):

- Sort the observations along X :

$$(X_{(1)}, Y_{(1)}), \dots, (X_{(n)}, Y_{(n)});$$

- Compute the ranks r_i of $Y_{(i)}$;
- Let $\xi(X, Y) = 1 - \frac{3}{n^2 - 1} \sum_i \|r_{i+1} - r_i\|$.

It converges to

$$\frac{\int \text{Var}_X \mathbb{E}_Y[\mathbf{1}_{Y \geq t} | X] dp_Y(t)}{\int \text{Var}_Y[\mathbf{1}_{Y \geq t}] dp_Y(t)}$$

which is 0 iff $X \perp\!\!\!\perp Y$ and 1 iff $Y = f(X)$ a.s. for some measurable f . If $X \perp\!\!\!\perp Y$, it is asymptotically Gaussian:

$$\sqrt{n} \xi_n(X, Y) \xrightarrow{d} N(0, 2/5).$$

Alternatives include:

- The maximum information coefficient (MIC): maximum of $\text{MI}(U; V) / \log \text{Min}(x, y)$, where $\text{MI}(U; V) = \text{KL}(p_{U,V} \| p_U p_V)$ and (U, V) is the discretization of the scatterplot of (X, Y) on an $x \times y$ grid, with $xy \leq n^{0.6}$;

- The distance correlation

$$a_{ij} = \|X_i - X_j\|$$

$$b_{ij} = \|Y_i - Y_j\|$$

$$A_{ij} = a_{ij} - a_{i\cdot} - a_{\cdot j}$$

$$B_{ij} = b_{ij} - b_{i\cdot} - b_{\cdot j}$$

$$\text{dCor} = \text{Cor}(A, B)$$

- HHG: for each pair (i, j) , $i \neq j$, compute the χ^2 statistic to test independence on the 2×2 contingency table for $|X_i - X| < |X_i - X_j|$ and $|Y_i - Y| < |Y_i - Y_j|$
- Hilbert-Schmidt independence criterion (HSIC)

$$\text{HSIC} = \frac{1}{n^2} \sum_{ij} k_{ij} \ell_{ij} + \frac{1}{n^4} \sum_{ijqr} k_{ij} \ell_{qr} - \frac{2}{n^3} \sum_{ijq} k_{ij} \ell_{iq}$$

$$k_{ij} = k(X_i, X_j)$$

$$\ell_{ij} = \ell(Y_i, Y_j)$$

$$k(x, y) = \ell(x, y) = \exp - \frac{1}{2} \left(\frac{x - y}{\sigma} \right)^2$$

R implementations in XICOR, energy (dCor), minerva (MIC), HHG, dHSIC.

On the power of Chatterjee's rank correlation

H. Shi et al.

Prefer Hoeffding's D , Blum-Kiefer-Rosenblatt's R , or Bergsma-Dassios-Yanagimoto's τ^* to Chatterjee's rank correlation.

$$D = \int [F(x, y) - F(x)G(y)]^2 dF(x, y)$$

$$R = \int [F(x, y) - F(x)G(y)]^2 dF(x) dF(y)$$

$$\begin{aligned} \tau^* = & 4I(x_1, x_3 < x_2, x_4; y_1, y_3 < y_2, y_4) \\ & + 4I(x_1, x_3 < x_2, x_4; y_2, y_4 < y_1, y_3) \\ & - 8I(x_1, x_3 < x_2, x_4; y_1, y_4 < y_2, y_3) \end{aligned}$$

where $I(x_1, x_2 < x_3, x_4) = \mathbf{1}_{\text{Max}(x_1, x_2) < \text{Min}(x_3, x_4)}$.

A simple measure of conditional dependence

M. Azadkia and S. Chatterjee

Conditional independence can be measured as

$$T(Y, Z|X) = \frac{\int \mathbb{E}_X \text{Var}_Z[P_Y(Y \geq t|Z, X)|X] dF_Y(t)}{\int \mathbb{E}_X \text{Var}_Y[\mathbf{1}_{Y \geq t}|X] dF_Y(t)}.$$

To estimate it:

- Let $N(i)$ be the index of the nearest neighbour of X_i ;
- Let $M(i)$ be the index of the nearest neighbour of (X_i, Z_i) ;
- Let $R_i = \sum_j \mathbf{1}_{Y_j \geq Y_i}$ be the rank of Y_i ;

$$T_n = \frac{\sum \text{Min}(R_i, R_{M(i)}) - \text{Min}(R_i, R_{N(i)})}{\sum R_i - \text{Min}(R_i, R_{N(i)})}.$$

It can be used for variable selection: add the variable X_j maximizing $T_n(Y, X_j|X_{j_1}, \dots, X_{j_k})$ until $T_n \leq 0$.

R implementation in FOCI.

The oracle estimator is sub-optimal for global minimum variance portfolio optimization
C. Bongiorno and D. Challet

Rotationally invariant estimators (RIE) of the variance matrix

$$\forall U \in O_n \quad \Sigma(U' \hat{S} U) = U' \Sigma(\hat{S}) U$$

filter the eigenvalues of the sample variance matrix \hat{S} while keeping its eigenvectors. The RIE optimal for the Frobenius distance (between the estimator and the (unknown) true variance matrix) is not optimal for minimum variance portfolio optimization.

A new parametrization of correlation matrices
I. Archakov and P.R. Hansen (2020)

A correlation matrix C can be parametrized with the off-diagonal elements of $\log C$: $C \mapsto \text{veclog } C$. This generalizes the Fisher transform:

$$\log \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \log(1 - \rho^2) & \frac{1}{2} \log \frac{1 + \rho}{1 - \rho} \\ \frac{1}{2} \log \frac{1 + \rho}{1 - \rho} & \frac{1}{2} \log(1 - \rho^2) \end{pmatrix}.$$

For covariance matrices, add $(\log \sigma_i)_i$.

To compute the inverse (without the diagonal elements of $\log C$), iterate $x \mapsto x - \log \text{diag } e^{A[x]}$, where $A[x]$ is A with its diagonal replaced by x , and return $C = e^{A[x]}$; we then have $\text{veclog } C = \text{vecl } A$.

Completing correlation matrices
O. Dreyer et al. (2021)

Complete concentration matrices by maximizing the entropy of the resulting Gaussian distribution. If the known correlations form a chordal graph, this can be done efficiently.

The Hurst roughness exponent and its model-free estimation
X. Han and A. Schied (2021)

The *standard Hurst exponent* measures the long-range dependence of a time series (or function), using its autocorrelation. The *Hurst roughness exponent* (the p th variation of x converges to 0 if $p > 1/H$ and to infinity if $p < 1/H$) measures the roughness of a function. They sometimes coincide (fBM), but not always.

The Gladyshev estimator is not scale-invariant, but can be made so.

Portfolio optimization with idiosyncratic and systemic risks for financial networks
Y. Yang et al. (2021)

Replace the variance matrix V with $\lambda \Sigma + (1 - \lambda)H$, where Σ is the matrix of covariances (V), or Kendall's τ 's, or tail dependence coefficients, and $H = (\sigma \sigma') \odot (CC')$ where $\sigma = \text{diag } V$ and C is the vector of clustering coefficients.

Portfolio optimization with options
J.R. Chan et al. (2021)

For portfolios of options, replace the variance matrix, in the Markowitz optimization problem, with

$$\Lambda_{ij} = \frac{P(O_i^+, O_j^+)}{P(O_i^+)P(O_j^+)},$$

where O_i^+ is the event “option i expires in the money” (it can be computed from the pairwise copulas of stock prices and the option strikes).

Online estimation and optimization of utility-based shortfall risk
A.S. Menon et al.

Utility-based shortfall risk (UBSR) generalizes CVaR (it is no longer coherent):

$$\text{SR}_\lambda(X) = \inf\{t \in \mathbf{R} : \mathbb{E}[U(t + x)] \geq -\lambda\}.$$

Mean-covariance robust risk measurement
V.A. Nguyen et al. (2021)

The *Chebychev ambiguity set* around a mean-covariance pair (μ, Σ) is the set of probability distributions with that expectation and that variance. The *Gelbrich distance* between mean-covariance pairs (μ_1, Σ_1) and (μ_2, Σ_2) is

$$\sqrt{\|\mu_1 - \mu_2\|^2 + \text{Tr}[\Sigma_1 + \Sigma_2 - 2(\Sigma_2^{1/2} \Sigma_1 \Sigma_2^{1/2})^{1/2}]}$$

It is the Wasserstein distance between $N(\mu_1, \Sigma_1)$ and $N(\mu_2, \Sigma_2)$.

The *Gelbrich ambiguity set* $G_\rho(\mu, \Sigma)$ is the set of distributions q whose mean-variance pair is at distance at most ρ of (μ, Σ) .

For a (translation-invariant, positive homogeneous, law-invariant) risk measure R ,

$$\sup_{q \in G_\rho(\mu, \Sigma)} R[-w'X] = -\mu'w + \alpha \sqrt{w' \Sigma w} + \rho \sqrt{1 + \alpha^2} \|w\|$$

where the risk coefficient α can be computed explicitly for VaR, CVaR, spectral risk measures, Kusuoka risk measures (supremums of spectral risk measures), distortion measures ($R[X] = \int_{\mathbf{R}} \tau dh(F_X(\tau))$ for some non-decreasing $[0, 1] \rightarrow [0, 1]$).

Optimal investment with risk controlled by weighted entropic risk measures
J. Xia (2021)

$$h(X) = \int_0^\infty \frac{1}{a} \mathbb{E}[e^{-aX}] d\mu(a)$$

Deep reinforcement learning for the optimal placement of cryptocurrency limit orders
M. Schnaubelt (2020)

Reinforcement learning (PPO) from limit order book features:

- Queue imbalance (difference between bid and ask volumes at various levels);
- Liquidity cost (cost of a market buy or sell order for 10, 20, 30, 50, 100 BTC or ETH);
- Volatility, drift, spread.

Deep partial hedging
S. Hou et al. (2021)

Partial hedging (hedging with less capital than needed, minimizing $P[\text{loss} > 0]$ (quantile hedging) or $E[\text{loss}|\text{loss} > 0]$ (efficient hedging)) is equivalent to Δ -hedging with a modified payoff.

Use a neural net to approximate that payoff, while accounting for transaction costs; also add a penalty $(\text{Min}_t V_t)_-$ to avoid being too much in the red.

A meta-method for portfolio management using machine learning for adaptive strategy selection
D. Kisiel and S. Gorse

Use xgboost to decide when to switch between hierarchical risk parity and $w_i \propto 1/\sigma_i^2$ (naive risk parity) on 18 ETFs. Features include:

- Returns, volatility, downside deviation, maximum drawdown of the strategies;
- Average return and volatility of the ETFs;
- Average and standard deviation of the correlations;
- Quality ratios: number of independent bets (square of the diversification ratio $\sum w_i \sigma_i / \sigma_{\text{port}}$ of the maximum diversification portfolio) divided by the number of assets;
- Non-parametric k -NN entropy estimator;
- Cophenetic correlation coefficient (correlation between distances and distances on a dendrogram);
- Standardized generalized variance: $\text{GV} = \det V$, $\text{SGV} = \det(V)^{1/p}$;
- Correlation matrix features: determinant, condition number, fraction of the variance explained by the eigenvalues outside the Marchenko-Pastur distribution.

Estimating security betas via machine learning
W. Drobetz et al.

Random forest (or neural network, or linear regression) to forecast realized beta (computed from daily returns over the next year) from stock characteristics (30 ratios, industries, historical (3m, 1y, 5y) betas).

A transformer-based model for default prediction in mid-cap corporate markets
K. Korangi et al. (2021)

Predict the term structure of default from

- Quarterly fundamental ratios,
- Quarterly financial ratios,
- Daily prices,

with a separate transformer for each, whose outputs are then combines, with the time series of default indicators (3m, 6m, 9m, 1y, 2y, 3y) as targets. Attention weights and Shapley contributions help interpret the model.

Market impact decay and capacity
H. Chan (2021)

When a trade is split over several days, subsequent days still feel the market impact of previous days – market impact does not immediately drop to zero but decays (it drops to zero after a month).

w_{it}^* : target weights

$$w_{it} = (1 - \tau)w_{i,t-1} + \tau w_{it}^*$$

τ : trading speed

$$\text{Impact} = \alpha \sqrt{\frac{\text{size}}{\text{ADV}}}, \quad \alpha = 100 \text{ bp}$$

$$\text{Decay}_t = \gamma^t, \quad \gamma = 0.85$$

$$\text{Cost}_{it} = C \Delta_{it} \sum_{s \leq t} \text{sign}_{is} \text{Decay}_{t-s} \text{Impact}_{is}$$

C : Capital

$$\text{Capacity} = \text{Max}\{C : \text{Max}_{\tau} \text{Sharpe}_{\tau} \geq \text{threshold}\}$$

The correlation risk premium: international evidence
G. Faria et al.

The correlation risk is the difference between the implied correlation

$$\text{IC} = \frac{\text{SV}_{\text{index}} - \sum_i w_i^2 \text{SV}_i}{\sum_{i \neq j} w_i w_j \sqrt{\text{SV}_i \text{SV}_j}}$$

SV = variance swap rates

and the realized correlation

$$\begin{aligned} \text{RC} = & \int_{S_t}^{\infty} \frac{2}{K^2} \left(1 - \log \frac{K}{S_t}\right) \text{Call}_{[t,T]}(K) dK \\ & + \int_0^{S_t} \frac{2}{K^2} \left(1 + \log \frac{K}{S_t}\right) \text{Put}_{[t,T]}(K) dK \end{aligned}$$

Similarly, the variance risk factor is the difference between the implied and the realized variance.

Sustainable investing in equilibrium
L. Pástor et al. (2020)

Green (ESG) have lower returns in equilibrium, but (by definition) positive returns when positive shocks hit the “ESG factor”.

**Deep differentiable reinforcement learning
and optimal trading**
T. Jaisson (2021)

Policy gradient, to trade one asset, with a 2-scale alpha: the optimal strategy uses the fast alpha to time the trades, and the slow alpha for the target weights.

**Time series simulation by conditional
generative adversarial net**
R. Fu et al.

To generate synthetic (univariate) time series, use a conditional GAN

$$\min_G \max_D \mathbb{E}_{x \sim \text{Data}} [\log D(x)] + \mathbb{E}_{x \sim \text{noise}} [\log(1 - DGx)]$$

or a WGAN

$$\min_G \max_D \mathbb{E}_{x \sim \text{Data}} [\log D(x)] - \mathbb{E}_{x \sim \text{noise}} [\log D(G(x))] \\ \|D\|_{\text{Lip}} \leq 1$$

with weight clipping, no batchnorm, 3 FC layers of 100+ nodes, leaky ReLU ($\alpha = .1$) activations and 30 inputs.

**qgam: Bayesian nonparametric
quantile regression modeling in R**
M. Fasiolo et al. (2021)

GAMs can be generalized to GAMs for location, scale and shape (GAMLSS) and quantile GAMs. To estimate a QGAM model, replace the pinball loss $\rho(z) = \tau z_+ + (1 - \tau)z_-$ with the smooth *extended log-f loss* (ELF)

$$\rho(z) = \lambda \log(1 + e^{z/\lambda\sigma}) - (1 - \tau) \frac{z}{\sigma}.$$

**JointAI: joint analysis and imputation
of incomplete data in R**
N.S. Erler et al. (2021)

Bayesian missing data imputation with MCMC (Gibbs sampling, via Jags) for GLMM (mixed GLM) and CLM (cumulative logit) multilogistic (mixed), survival models.

Informed Bayesian inference for the A/B test
Q.F. Gronau et al. (2021)

abtest: A/B testing for 3 hypotheses (positive, negative, or no effect), with a prior (expert knowledge).

**BFPack: flexible Bayes factor
testing of scientific theories in R**
J. Mulder et al. (2021)

Bayesian statistical tests for multiple hypotheses involving inequalities and equalities (e.g., $\mu = 0$ vs $\mu > 0$ vs $\mu < 0$, or $\sigma_1 = \sigma_2 < \sigma_3$ vs $\sigma_1 < \sigma_2 = \sigma_3$ vs $\sigma_1 = \sigma_2 = \sigma_3$ vs neither).

**Shrinkage in the time-varying parameter model
framework using the R package shrinkTVP**
P. Knaus et al. (2021)

MCMC estimation of the Bayesian time-varying parameter (TVP) model

$$y_t = x_t \beta_t + \varepsilon_t \\ \beta_t = \beta_{t-1} + w_t$$

with a normal-gamma-normal (NGG) prior (normal gamma, horseshoe, and lasso are limiting cases).

**Modeling univariate and multivariate
stochastic volatility in R
with stochvol and factorstochvol**
D. Hosszejni and G. Kastner (2021)

Efficient (C++) MCMC estimation of (uni- or multivariate) non-Gaussian, asymmetric (leverage effect) stochastic models (non-linear state space models).

**dalmatian: a package for fitting
double hierarchical linear models
in R via Jags and Nimble**
S. Bonner et al. (2021)

Mixed model, with fixed and random effects for both mean and variance.

**Machine learning optimization algorithms
and portfolio allocation**
S. Perrin and T. Roncalli (2019)

Four machine learning algorithms are applicable to portfolio optimization (e.g., with a Herfindahl penalty to increase diversification): coordinate descent, ADMM, proximal gradient, and Dykstra's algorithm.

Coordinate descent optimizes one variable at a time, keeping the others fixed; those 1-dimensional optimization problems often have an explicit solution (e.g., soft-thresholding, for the lasso) or can be solved with line search.

ADMM solves problems of the form

$$\begin{array}{ll} \text{Find} & x, y \\ \text{To minimize} & f(x) + g(y) \\ \text{Such that} & Ax + By = c \end{array}$$

by iterating

$$\begin{aligned} x &\leftarrow \underset{x}{\text{Argmin}} f(x) + \lambda \|Ax + By - c + u\|_2^2 \\ y &\leftarrow \underset{y}{\text{Argmin}} g(y) + \lambda \|Ax + By - c + u\|_2^2 \\ u &\leftarrow u + (Ax + By - c). \end{aligned}$$

Finding a good value of λ is tricky:

- Increase λ if $\|r\| \gg \|s\|$,
- Decrease λ if $\|r\| \ll \|s\|$,

where $r = Ax + By - c$ and $s = 2\lambda A'B\Delta y$.

Convergence to a moderately accurate solution is fast; convergence to a more accurate solution much slower.

To put an optimization problem in ADMM form, consider the following “tricks”:

- To minimize $f(x) + g(x)$, minimize $f(x) + g(y)$ subject to $x = y$;
- To minimize $f(x)$ subject to $x \in \Omega$, minimize $f(x) + \mathbf{1}_\Omega(y)$ subject to $x = y$;
- To minimize $f(x) + g(x)$ subject to $x \in \Omega_1 \cap \Omega_2$, minimize $f(x) + \mathbf{1}_{\Omega_1}(x) + g(y) + \mathbf{1}_{\Omega_2}(y)$ subject to $x = y$;
- To minimize $f(x) + g(Ax + b)$, minimize $f(x) + g(y)$ subject to $y = Ax + b$.

The **proximal operator** is

$$\text{prox}_f(v) = \underset{x}{\text{Argmin}} f(x) + \frac{1}{2} \|x - v\|_2^2;$$

it generalizes the Euclidean projection (for $f = \mathbf{1}_\Omega$) and can often be computed explicitly.

There is no formula for $\text{prox}_{f_1+f_2}$, but it can be computed by iterating

$$\begin{aligned} x &\leftarrow \text{prox}_{f_1}(y + p) \\ p &\leftarrow y + p - x \\ y &\leftarrow \text{prox}_{f_2}(x + q) \\ q &\leftarrow x + q - y \end{aligned}$$

starting with $x = y = v$ and $p = q = 0$. **Dykstra’s algorithm** generalizes this to a sum of m terms. It can be used to project on an intersection: $\mathbf{1}_{\Omega_1 \cap \dots \cap \Omega_m} = \mathbf{1}_{\Omega_1} + \dots + \mathbf{1}_{\Omega_m}$, e.g., an intersection of half-spaces (*i.e.*, general inequality constraints, $Ax = b$, $Cx \leq d$).

Dykstra’s algorithm, ADMM and coordinate descent: connections, insights, and extensions
R.J. Tibshirani (2017)

The best approximation problem

$$\begin{aligned} \text{Find} \quad & u \in \mathbf{R}^n \\ \text{To minimize} \quad & \|y - u\|^2 \\ \text{Such that} \quad & u \in C_1 \cap \dots \cap C_d \end{aligned}$$

can be solved with Dijkstra’s algorithm, iterating

$$\begin{aligned} u &\leftarrow P_{C_k}(u + z_k) \\ z_k &\leftarrow z_k - \Delta u \end{aligned}$$

with a different k at each iteration (use a different dual variable for each C_k).

Its dual is the regularized regression problem

$$\begin{aligned} \text{Find} \quad & w \in \mathbf{R}^p \\ \text{To minimize} \quad & \frac{1}{2} \|y - Xw\|^2 + \sum_i h_i(w_i) \end{aligned}$$

and the coordinate descent algorithm is equivalent to Dykstra’s.

A note on portfolio optimization with quadratic transaction costs
P. Chen et al. (2019)

The transaction costs should be included in the budget constraint: for linear costs, $w'\mathbf{1} = 1$ becomes $w'\mathbf{1}_\tau \|w - w_0\|_1 = 1$ – and we should also distinguish between bid and ask prices. For quadratic transaction costs, the problem is no longer convex, but still amenable to ADMM.

[$w'\mathbf{1} + C(w, w_0) = 1$ is not convex, but $w'\mathbf{1} + C(w, w_0) \leq 1$ is.]

Improving the robustness of trading strategy backtesting with Boltzmann machines and generative adversarial networks

Use a convolutional Wasserstein GAN to generate synthetic time series.

Portfolio optimization with sparse multivariate modelling
P.F. Procacci and T. Aste (2021)

Use the TMFG-LoGo variance matrix for portfolio optimization (the sparsity structure of Σ^{-1} is given by the TMFG-filtered graph, and the correlation of TMFG-linked assets is preserved).

Sparse causality network retrieval from short time series
T. Aste and T. Di Matteo

An information filtering approach to stress testing: an application to FTSE markets
I. Seabrook et al. (2021)

Given a multivariate Gaussian

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N \left[\begin{pmatrix} \mu_X \\ \mu_Y \end{pmatrix}, \begin{pmatrix} \Omega_{XX} & \Omega_{XY} \\ \Omega_{YX} & \Omega_{YY} \end{pmatrix} \right],$$

the impact of X on Y is the average change of the variables in Y for a unit change of the variables in X

$$\begin{aligned} L_{X \rightarrow Y} &= \underset{i}{\text{Mean}} (E[Y_i | X = \mu_X + 1] - E[Y_i]) \\ &= \frac{1}{n_Y} \mathbf{1}'_Y \Omega_{YX} \Omega'_{XX} \mathbf{1}_X. \end{aligned}$$

Make Ω^{-1} sparse with TMFG filtering.

Cluster the dates into market states, with a log-likelihood penalized for frequent cluster changes (Viterbi algorithm).

The most central groups (industrial sectors) have a higher response but a lower impact, because of the number of links within those groups rather than their centrality.

***Simplicial persistence of financial markets:
filtering, generative processes
and portfolio risk***
J.D. Turiel et al. (2020)

Check which triangles, separators, tetrahedra of the TMFG (triangulated maximally filtered graph) are persistent, *i.e.*, present at times t and $t + \tau$ (and the number of such triangles as τ varies).

***A universal end-to-end approach
to portfolio optimization via deep learning***
C. Zhang et al. (2021)

End-to-end portfolio construction, from daily price data, with a few constraints, *e.g.*, leverage, maximum weight, number of assets (with a sorting operation – it is differentiable).

Deep learning for portfolio optimization
Z. Zhang et al.

Only 4 assets (ETFs), with an LSTM trained on 10 years of daily price data.

***End-to-end risk budgeting portfolio
optimization with neural networks***
A.S. Uysal et al. (2021)

End-to-end portfolio construction, from 10 years of daily price data, for 7 assets, with or without an optimization layer (risk budgeting).

***Integrating prediction
in mean-variance portfolio optimization***
A. Butler and R.H. Kwon (2021)

End-to-end portfolio optimization has a quadratic optimization reformulation in the special case where the return forecasts are a linear combination of the features. Application to 24 commodities, with trend and carry as features.

Scenario-based risk evaluation
R. Wang and J.F. Ziegel

The conditional value at risk (VaR) and expected shortfall (ES) of X given the market state S are

$$\begin{aligned}\text{CoVaR}_{S|X} &= \text{VaR}[X | S = \text{VaR}_S] \\ \text{ES}_S X &= \text{E}[X | X \leq \text{CoVaR}_S X].\end{aligned}$$

More generally, given scenarios Q_1, \dots, Q_n , one can define

$$\begin{aligned}\text{MES } X &= \text{Max}_i \text{ES}^{Q_i} X \\ \text{MVaR } X &= \text{Max}_i \text{VaR}^{Q_i} X \\ \text{AES } X &= \text{Mean}_i \text{ES}^{Q_i} X \\ \text{iMES } X &= \text{Mean}_{q \in [p, 1]} \text{MVaR}_q X \\ \text{rMES } X &= \text{ES}[\text{Max}_i X_i], \quad X_i \underset{\text{indep.}}{\sim} Q_i.\end{aligned}$$

AES and eMES are coherent and comonotonic additive.

Basel III and IV require a *stress adjustment*, ES calibrated on the worst 12 months (MES), and a *dependence adjustment*, average of the ES calibrated on the worst 12 months and that calibrated on the worst possible dependence structure (comonotonic risk factors: compute an ES separately for each category of risk factors and add them).

***A framework
for measures of risk under uncertainty***
T. Fadina et al.

(Coherent) risk measures are computed (in the case of a portfolio) from portfolio weights and a distribution of asset retruns. A generalized risk measure uses, instead, a *set* of distributions on asset returns, *i.e.*, a set of scenarios. One can then compute the worst case (or average, etc.) VaR, ES, etc.

Forecasting market states
P.F. Procacci and T. Aste

Markov switching model (Viterbi algorithm) for the precision matrix, estimated with the TMFG-LoGo network filtering approach (intuitively, k -means, with the Mahalanobis distance, and a penalty for cluster changes).

***Toeplitz inverse covariance-based clustering
of multivariate time series data***
D. Hallac et al. (2017)

To estimate the correlation matrix between $\mathbf{x}_t, \dots, \mathbf{x}_{t+k}$, use the graphical lasso, but impose a Toeplitz constraint to ensure $\text{Cor}(\mathbf{x}_t, \mathbf{x}_{t+\ell}) = \text{Cor}(\mathbf{x}_s, \mathbf{x}_{s+\ell})$.

***The adoption of blockchain-based
decentralized exchanges***
A. Capponi and R. Jia (2021)

The pricing function $F : \mathbf{R}^2 \rightarrow \mathbf{R}^2$ of an automated market maker (AMM) satisfies

$$\begin{aligned}F_x, F_y &> 0 \\ F_{xx}, F_{yy} &< 0 \\ F_{xy} &> 0 \\ F(\lambda x, \lambda y) &= \lambda^k F(x, y), \quad k > 0 \\ F_x/F_y &\xrightarrow{x \rightarrow 0} \infty \\ F_x/F_y &\xrightarrow{x \rightarrow \infty} 0 \\ F_x/F_y &\xrightarrow{y \rightarrow 0} 0 \\ F_x/F_y &\xrightarrow{y \rightarrow \infty} \infty.\end{aligned}$$

The number of assets in the AMM (usually 2), and the volatility of their exchange rates, increase arbitrage opportunities and lowers the incentives for liquidity

providers. Higher curvature mitigates arbitrage, but benefits more investors than liquidity providers.

G-learner and GIRL: goal-based wealth management with reinforcement learning
M.F. Dixon and I. Halperin (2020)

G-learning is Q-learning (off-policy RL) with an entropy penalty $H(\pi)$ or $KL(\pi||\pi_0)$. The *free energy* F is the entropy-regularized state value function. The *G-function* is the entropy-regularized state-action value function.

With a few (Gaussian) assumptions and (quadratic) approximations, the problem is tractable.

AlphaPortfolio: direct construction through deep reinforcement learning and interpretable AI
L.W. Cong (2019)

Train a neural network, computing portfolio weights, from 12 months of 50 features (price, investments, profitability, intangibles, value, liquidity) for 3000 stocks, over 50 years, on 2-year mini-batches, to directly minimize the information ratio. One can add interactions with the environment:

- Trading costs;
- Trade impact;
- Fund failure (loss beyond 50% within 1 year).

The portfolio is Q stocks long, Q stocks short, with exponentially decaying weights.

The network could be an LSTM with attention

$$\begin{aligned} h_k &= \text{LSTM}(h_{k-1}, x_k) \\ r &= \sum_k \text{ATT}(h_K, h_k) h_k \\ \text{ATT}(h_K, h_k) &= \frac{\exp \alpha_k}{\sum_\ell \exp \alpha_\ell} \\ \alpha_k &= W'_0 \tanh(W_1 h_k + W_2 h_K) \end{aligned}$$

or a transformer (concatenation of several attention blocks)

$$\begin{aligned} V, K, Q &= W_0 x, W_1 x, W_2 x \\ \text{Attention}(Q, K, V) &= \text{Softmax} \left(\frac{QK'}{\sqrt{d_k}} \right) V \end{aligned}$$

(here, $\sqrt{d_k}$ helps keep the variance around 1).

Each asset is processed separately, and the results are combined with a *cross-asset attention network*

(CAAN)

$$\begin{aligned} q_i, k_i, v_i &= W_Q r_i, W_K r_i, W_V r_i \\ \beta_{ij} &= \frac{q'_i k_j}{\sqrt{d_k}} \\ a_i &= \sum_j \text{SATT}(q_i, k_j) v_j \\ \text{SATT}(q_i, k_j) &= \frac{\exp \beta_{ij}}{\sum_\ell \exp \beta_{i\ell}} \\ s_i &= \tanh(w' a_i + e). \end{aligned}$$

To interpret the model:

- Take the features with the largest $E \left[\left| \frac{\partial \text{IR}(x)}{\partial x_i} \right| \right]$ and use them in a polynomial regression with a lasso penalty;
- Alternatively, use “textual factors” (topics) from SEC filings to reproduce (distill) the model outputs.

Multi-horizon forecasting for limit order books: novel deep learning approaches and hardware acceleration using intelligent processing units
Z. Zhang and S. Zohren (2021)

Sequence-to-sequence (seq2seq) model, with attention, for multi-step forecasts of price changes (direction or value) from limit order book (LOB) data). Process the LOB (bid, ask, volume, price) as if it were an image, with inception modules.

input	x_t
encoder	$h_i = f(h_{i-1}, x_i)$
context	$c_t = \sum_i \alpha_{it} h_i$
decoder	$h'_t = f(h'_{t-1}, t_{t-1}, c_t)$
output	$y_t \sim g(h'_t, c_t)$
attention weights	$\alpha_{it} \propto \exp h_i^\top h'_{t-1}$ or $\exp h_i^\top W h'_{t-1}$ or $\exp \tanh W[h_i; h'_{t-1}]$
	i : time (past)
	t : time (future)

Textual factors: a scalable, interpretable, and data-driven approach to analyzing unstructured information
L.W. Cong et al. (2018)

Combine word embeddings and topic models:

- Compute word2vec embeddings
- Group the words into clusters, using LSH (locality-sensitive hashing);
- Assume that the topics are the clusters, and estimate a topic model, one topic at a time.

Applications:

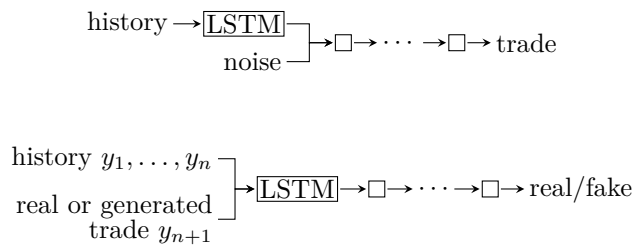
- Macroeconomic (CPI, GDP, housing prices, unemployment, S&P, investment) forecasts from news (WSJ);
- VIX back-filling (1889 to 1986);
- Model interpretation.

Forest through the trees: building cross-sections of stock returns
S. Bryzgalova et al. (2019)

Instead of double and triple sorts (decision trees, to forecast or explain future returns, using the same variable and threshold for all nodes at the same depth), use a more general “asset pricing tree”.

Towards realistic market simulations: a generative adversarial networks approach
A. Coletta et al. (2021)

Train a cGAN (WGAN-GP) to generate new orders mimicking real orders, and feed them (as an aggregated agent) to a market simulator (ABIDES) on which you can train new trading strategies.



Zero-liquidation loans: a structured product approach to DeFi lending
A. Sardon (2021)

A zero-liquidation loan is a loan, say of USD, with an ETH collateral, at whose expiry the borrower can choose to repay the loan in USD or ETH (*i.e.*, leave the ETH collateral to the lender).

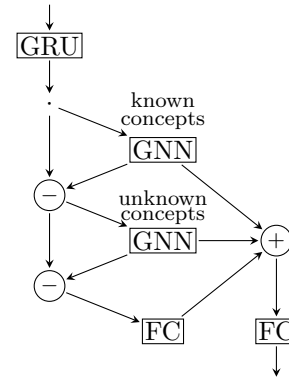
Hist: a graph-based framework for stock trend forecasting via mining concept-oriented shared information
W. Xu et al.

Forecast future stock returns from:

- Stock features,
- Known “concepts” (sectors, industries, regions, etc.),
- Unknown concepts (initialize them with stock embeddings; link each stock to the nearest concept, excluding its own; discard unlinked concepts).

The GNNs (graph neural networks) are applied to the

bipartite stock-concept graphs.



Embracing advanced AI/ML to help investors achieve success: RL for financial goal planning
S. Mohammed et al.

RL (DQN, OpenAI Gym) for financial planning [not unlike what I did at SRLGlobal]:

- Action: proportion of the wealth used for consumption, each year, discretized;
- State: tax rate, life expectancy, wealth, market state;
- Environment: Monte Carlo simulations;
- Reward: piecewise linear function of the probability of success.

FinEAS: Financial embedding analysis of sentiment
A. Gutiérrez-Fandiño et al. (2021)

BERT provides good token embeddings, but poor sentence embeddings: prefer *sentenceBERT*, a pre-trained BERT model (masked language model, trained for next-sentence prediction), fine-tuned on an entailment task. For sentiment analysis, either use sentenceBERT for feature extraction, or fine-tune it.

MAD risk parity portfolios
C. Ararat et al. (2021)

The mean absolute deviation

$$\text{MAD}(X) = \mathbb{E} |X - \mathbb{E} X|$$

is a *deviation measure*; the corresponding risk measure is

$$\rho(X) = \mathbb{E} [-X + |X - \mathbb{E} X|].$$

The MAD is not differentiable, but it has subgradients; Euler’s theorem is still valid:

$$\forall s \in \partial \text{MAD}(w) \quad \text{MAD}(w) = s'w.$$

A MAD-risk-parity portfolio w is such that

$$\exists s' \in \partial \text{MAD}(w) \quad \forall i, j \quad s_i w_i = s_j w_j.$$

For long-only portfolios: $w_i \propto \text{MAD}(x_i)^{-1}$.

Forecasting financial market structure from network features using machine learning
D. Castilho et al.

Take the current correlation matrix (on 100 assets), turn it into a graph (e.g., the minimum spanning tree, or edges for correlations in the top quartile, or above some fixed threshold), compute node and edge features (centralities, similarities, etc.) and use them to forecast the next correlation matrix (or graph).

Hierarchical information clustering by means of topologically embedded graphs
W.M. Song et al.

- Compute a correlation matrix;
- Build its PMFG;
- Find the non-separating 3-cliques, *i.e.*, the 3-cliques separating the graph into a (non-empty) interior and a (non-empty) exterior;
- The interior and the clique form a planar graph (“bubble”); the exterior and the clique form another planar graph; they can be processed, recursively, in the same way;
- We obtain a tree, with the bubbles as nodes, and the cliques as edges;
- Orient the edges by comparing the weights of the edges between clique and bubble;
- Converging bubbles define clusters.

Data considerations in graph representation learning for supply chain networks
A. Aziz et al. (2021)

To forecast missing edges in the supply chain, use a GNN on a knowledge graph, with several node types (company, country, capability, product, certification) and edge types (supplier, located, produces, complementary product, etc.), after completing the knowledge graph using cooccurrence frequencies.

A machine learning approach for predicting hidden links in supply chain with graph neural networks
E.E. Kosasih and A. Brintrup

For each pair of nodes x, y , consider the union of their (1-hop?) egonets, colour the nodes with the double radius distance $f(z) = [d(z, x), d(z, y)]$, and feed the corresponding 1-hot embedding to a GNN to predict the presence or absence of an edge x – y (there is one such graph for each (x, y) pair).

Previous approaches to link prediction relied on node similarity (but this assumes homophily – similar companies are more likely to be competitors than supplier/customer), maximum likelihood, or hand-crafted features.

General compound Hawkes processes for mid-price prediction
M. Sjogrena and T. DeLise (2021)

Model mid-prices as a compound Hawkes process

$$S_t = S_0 + \sum_{k=0}^{N(t)} X_k$$

where N is a Hawkes process and X an n -state Markov chain (e.g., taking values $\{\pm\delta\}$, or $\{-a, +b\}$, etc.).

Data-driven hedging of stock index options via deep learning
J. Chen and L. Li (2021)

To choose the hedge rate δ of an option, fit a

- Non-linear, but parametric model, combining Black-Scholes (BS) delta and vega (Hull-White);
- Non-linear (splines) model, combining moneyness, BS delta, and time-to-maturity;
- Non-linear model (neural network), combining moneyness, BS delta, time-to-maturity, VIX level (for calls), index returns (for puts) – using a GRU or LSTM for the VIX and the index returns does not improve things.

Deep reinforcement learning for active high-frequency trading
A. Briola et al.

PPO (stable baselines + SMBO for hyperparameter optimization) for single-stock intraday trading, from LOB data. The state includes the volumes of the first 10 levels for the past 10 ticks, the agent’s position, the mark-to-market value of its position, the current bid-ask spread.

FinRL: a deep reinforcement learning library for automated stock trading in quantitative finance
X.Y. Liu et al. (2020)

Media abnormal tone, earnings announcements, and the stock market
D. Ardia et al. (2021)

To estimate the “abnormal tone” present in news (*news contextualize information*):

- Compute the tone of the news;
- Subtract the average for other companies in the universe;
- Aggregate around events;
- Remove the effect of B/P , size, and earnings call tone.

A model of text for experimentation in the social sciences
M.E. Roberts et al.

Topic model whose weights are parametrized by observed variables.

**Clustering market regimes
using the Wasserstein distance
B. Horvath et al. (2021)**

To cluster a time series into regimes, use the Wasserstein k -means algorithm on time series windows (seen as empirical distributions).

**Measuring geopolitical risk
D. Caldara and M. Iacoviello (2021)**

The geopolitical risk (GPR) index counts the number of words or articles, in 10 US, UK, Canadian newspapers, related to war threats, peace threats, military build-up, nuclear threat, terrorist threat, beginning of war, escalation of war, terrorist attacks, with a few excluded words (“movie”, “obituary”, “cancer”, etc.). Add country and city mentions for country-specific measures.

Dictionary and data available.

**Online estimation and optimization
of utility-based shortfall risk
A.S. Menon et al.**

Utility-based shortfall risk (UBSR) generalizes CVaR (it is no longer coherent):

$$\text{SR}_\lambda(X) = \inf \{ t \in \mathbf{R} : \mathbb{E}[U(t+x)] \geq -\lambda \}.$$

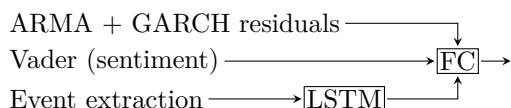
**Risk measures beyond frictionless markets
M. Arduca and C. Munari (2021)**

Many risk measures can be defined as the minimum amount x to invest in some asset $(S_t)_t$ (e.g., cash, $S_t \equiv 1$), to make the strategy X acceptable (e.g., $\mathcal{A} = \{X : q_\alpha(X) \geq 0\}$).

$$\rho(X) = \int \{ x S_0 : X + x S_1 \in \mathcal{A} \}$$

This can be generalized to more reference assets $S_{i,t}$, or even strategies on those assets.

**Forecasting crude oil price
using event extraction
J. Liu and X. Huang (2021)**



**Exploration of the parameter space
in macroeconomic agent-based models
K. Naumann-Woleske et al.**

Macroeconomic agent-based models (ABM) have many parameters, but only changes in a few directions, in parameter space, actually influence the output. These directions (eigenvectors of the Hessian for the largest eigenvalues) are not axis-aligned.

**Investor sentiment in the stock market
M. Baker and J. Wurgler**

Define a sentiment index as the first principal component of several sentiment proxies:

- Trading volume;
- Dividend premium;
- Closed-end fund discount;
- Number of IPOs;
- First day return on IPOs;
- Equity share in new issues

and also VIX, mutual fund flows, insider trading, etc.

**Non-asymptotic estimation of risk measures
using stochastic gradient Langevin dynamics
J. Chu and L. Tangpi (2021)**

The expected shortfall can be computed as the solution of an optimization problem,

$$\text{ES}_u X = \inf_{q \in \mathbf{R}} \frac{1}{1-u} \mathbb{E}[(X-q)^+] + q$$

For $X = f(r, S)$, where $S = (X_0, \dots, X_n)$ is a set of risk factors and r the parameters of the strategy f , e.g., $f(r, S) = \sum r_i S_i$ for a portfolio, it can be computed with gradient descent.

**Heterotic risk models
Z. Kakushadze (2015)**

Hierarchical industry classifications define nested factor models (for the correlation matrix, not the variance)

$$V_0 = \beta_{01} V_1 \beta'_{01} + \Delta_0$$

$$V_1 = \beta_{12} V_2 \beta'_{12} + \Delta_1$$

$$V_2 = \beta_{23} V_3 \beta'_{23} + \Delta_2$$

(where 0, 1, 2, 3 are stocks, subindustries, industries, sectors).

**ETF risk models
Z. Kakushadze and W. Yu (2021)**

To compute the variance matrix of ETFs, estimate a “heterotic risk model” after building an ETF taxonomy:

- Largest sector, industry, subindustry (if one dominates, othersize “broad”);
- Largest capbin;
- Largest region or country;
- “Value” or “growth” or “mixed”;
- Bond type
- Duration bin;
- Credit rating
- Commodity type;
- “Leveraged”, “inverse”, “normal”.

Statistical industry classification
Z. Kakushadze and W. Yu (2016)

Cluster *normalized returns* (not raw returns, or correlations), after “smoothing” stocks with an unreasonably small volatility.

Use the *effective rank* as the number of clusters.

If you already have an industry classification, it is likely better: only apply the clustering inside large subindustries.

Cryptoasset factor models
Z. Kakushadze (2018)

(1-day) momentum, intraday volatility, and size – volume has little effect.

4-factor model for overnight returns
Z. Kakushadze (2015)

Size, intraday volatility (or $(H - L)/C$, averaged over 21 days), momentum, liquidity (volume).

Momentum residual networks
M.E. Sander et al. (2021)

Residual networks are discretizations of first order differential equations, $x_{n+1} = x_n + f_\theta(x_n)$. Momentum residual networks are discretizations of second-order ODEs.

$$v_{n+1} = \gamma v_n + (1 - \gamma)f(x_n)$$

$$x_{n+1} = x_n + v_{n+1}$$

$$\varepsilon \ddot{x} + \dot{x} = f(x), \quad \varepsilon = \frac{1}{1 - \gamma}$$

They are invertible.

Loss surface simplexes
for mode-connecting volumes
and fast ensembling
G.W. Benton et al.

Global minima are believed to be connected by multi-dimensional submanifolds. Search for simplicial complexes $K = \text{chull}(x_1, \dots, x_n)$ minimizing

$$\mathbb{E}_{x \sim \text{Unif}(K)} [\ell(x)] - \lambda \log \text{volume}(K)$$

where the expected loss in the simplex is approximated from 5 random points in it, and the number of vertices, n , progressively increases.

GRAND: graph neural diffusion
B.P. Chamberlain et al. (2021)

GNNs are equivalent to the explicit, single-step Euler discretization of a diffusion PDE: multi-step (Runge-

Kutta) and implicit schemes perform better.

x : node features (scalar field)

\mathcal{X} : edge features (vector field)

$$\langle x, y \rangle = \sum_i x_i y_i$$

$$\langle\langle \mathcal{X}, \mathcal{Y} \rangle\rangle = \sum_{i>j} w_{ij} \mathcal{X}_{ij} \mathcal{Y}_{ij} \quad w_{ij} \in \{0, 1\}$$

$$\mathcal{X}_{ij} = -\mathcal{X}_{ji} \quad \text{alternating edge field}$$

$$(\nabla x)_{ij} = x_j - x_i$$

$$(\text{div } \mathcal{X})_i = \sum_{j:(i,j) \in E} \mathcal{X}_{ij} = \sum_j w_{ij} \mathcal{X}_{ij}$$

$$\langle\langle \nabla x, \mathcal{X} \rangle\rangle = \langle x, \text{div } \mathcal{X} \rangle \quad \text{adjunction}$$

The diffusion equation

$$\frac{\partial x}{\partial t} = \text{div}(G(x) \nabla x), \quad G_{ij,ij} = a(x_i, x_j),$$

for some attention function a , is of the form

$$\frac{\partial x_t}{\partial t} = (A(x_t) - I)x_t = \bar{A}(x_t)x_t.$$

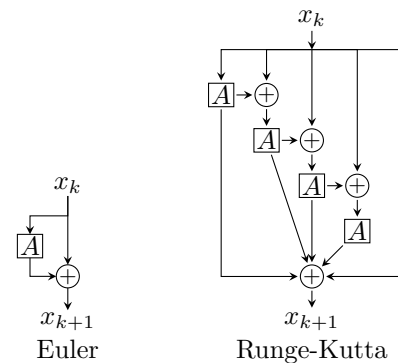
The explicit scheme is

$$x_i \leftarrow x_i + \sum_j a_{ij}(x_j - x_i),$$

i.e., $\mathbf{x}_{k+1} = (I + \tau A_k) \mathbf{x}_k$. The implicit scheme is $(I - \tau A_k) \mathbf{x}_{k+1} = x_k$ (it requires solving a linear system). Multi-step schemes are of the form

$$\sum_{j=0}^s \alpha_j x_{k+j} = \tau \sum_{s=0}^s \beta_j \bar{A}(x_{k+j}) x_{k+j}.$$

There are also adaptive schemes.



For the attention matrix A , either learn it (linear PDE), or use

$$A(X_i, X_j) = \text{Softmax} \frac{(W'_k X_i)' (W'_Q X_j)}{d_k}$$

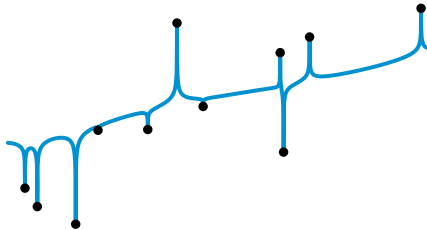
where W_K and W_Q are learned.

Try to re-wire the graph after each backward step by thresholding the attention.

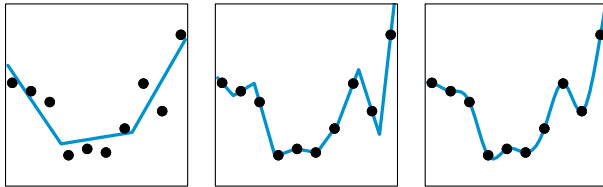
Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation
M. Belkin

Interpolating models, which overfit the data, may perform well out of sample. Examples include:

- k -nearest neighbour with a singular kernel, $k(x, z) = \|x - z\|^{-\alpha}$, $\alpha > 0$ (or a logarithmic kernel)



- Random ReLU features, $f_w(x) = \sum w_k(a_k x + b_k)_+$;



- Random Fourier features

$$f_w(x) = \sum_{k=1}^m w_k e^{i\langle v_k, x \rangle}$$

$$v_k \stackrel{\text{iid}}{\sim} N(0, I) \quad \text{fixed}$$

$$w = \underset{w: \forall i f_w(x_i) = y_i}{\text{Argmin}} \|w\|$$

```
## Error in download.file(paste0(url, fn),
## destfile): download from 'https://www2.harvardx.harvard.edu/courses/IDS_08_v2_03/train-images-idx3-ub
failed
## Error in Y[, i + 1] <- y == i: replacement
has length zero
## Error in matrix(rnorm(dim(x)[2] * k), nr
= dim(x)[2], nc = k): non-numeric matrix
extent
## Error in plot.window(...): need finite
'ylim' values
## Error in axis(1): CreateAtVector [log-axis()]:
exp[0] = 0 < 0!
## Error in plot.window(...): need finite
'ylim' values
## Error in axis(1): CreateAtVector [log-axis()]:
exp[0] = 0 < 0!
## Error in plot.window(...): need finite
'ylim' values
## Error in axis(1): CreateAtVector [log-axis()]:
exp[0] = 0 < 0!
```

- Linear regression $\beta = X^\dagger y$.

In the under-parametrized regime, we minimize the loss. In the over-parametrized regime, we minimize some penalty (it does not really matter which one) under the constraint that the loss be zero. The test error exhibits a “double descent” pattern.

Since global minima form a (potentially curved) sub-manifold, the problem is not locally convex. Instead, the *PL condition* suffices for gradient descent convergence:

$$\forall w \quad \frac{1}{2} \|\nabla \ell(w)\|^2 \geq \mu [\ell(w) - \ell(w^*)].$$

It is non-local, but may be replaced by PL^* , locally:

$$\frac{1}{2} \|\nabla \ell(w)\|^2 \geq \mu \cdot \ell(w)$$

with $\mu = \lambda_{\min}(k)$, where k is the tangent kernel:

$$k_{x,y}(w) = \langle \nabla_w f(x), \nabla_w f(y) \rangle.$$

Decoupled weight decay regularization

I. Loshchilov and F. Hutter (2019)

Gradient descent $\theta \leftarrow \theta - \alpha \nabla f(\theta)$ can be regularized with weight decay

$$\theta \leftarrow (1 - \lambda)\theta - \alpha \nabla f(\theta).$$

For SGD, this is equivalent to L^2 regularization; for Adam, it is not: weight decay works, L^2 regularization does not – do not put it in the loss function, keep it separate.

Lookahead optimizer:

k steps forward, 1 step back

M.R. Zhang et al. (2019)

Iterate:

- Run the optimizer for k steps, from $\phi_0 = \theta_t$ to ϕ_k ;
- Update θ in the direction of ϕ_k : $\theta_{t+1} = \phi_0 + \alpha(\phi_k - \phi_0)$.

There is a principled way of choosing α .

Unsupervised deep embedding for clustering analysis

J. Xie et al. (2016)

Jointly learn a low-dimensional data representation and a clustering:

- Initialize the data representation with an autoencoder $x \mapsto z = f_\theta(x)$;

- Initialize the cluster centroids $(\mu_j)_{1 \leq j \leq k}$ (k known), e.g., with k -means on the z_i 's';
- Measure the similarity between point z_i and centroid μ_j with a t distribution (as with t -SNE) with $\alpha = 1$ degrees of freedom,

$$q_{ij} \propto (1 + \alpha^{-1} \|z_i - \mu_j\|^2)^{(\alpha+1)/2}, \quad \sum_j q_{ij} = 1;$$

- As in self-training, define target distributions using the high-confidence predictions,

$$p_{ij} \propto q_{ij}^2 / f_j, \quad f_j = \sum_i q_{ij};$$

- Jointly optimize θ and μ to minimize

$$\sum_i \text{KL}(p_i \| q_i) = \sum_{ij} p_{ij} \log \frac{p_{ij}}{q_{ij}}.$$

**MLP-Mixer:
an all-MLP architecture for vision**
I. Tolstikhin et al.

The MLP mixer processes images by splitting them into patches and successively applying two types of layers:

- MLP on the patch dimension (the same for all channels and offsets (columns));
- MLP on the offset and channel dimensions (the same for each patch (row)).

Patches are all you need

Replace the MLP operations in MLP-mixer with convolutions.

Liquid time-constant networks
R. Hasani et al. (2021)

There are many variants of neural ODEs:

NODE	$\frac{dx_t}{dt} = f_\theta(x_t, I_t, t)$
CT-RNN	$\frac{dx_t}{dt} = -\frac{x_t}{\tau} + f_\theta(x_t, I_t, t)$
LTC	$\frac{dx_t}{dt} = -\frac{x_t}{\tau} + f_\theta(x_t, I_t, t)(A - x_t)$

The effective time scale is no longer a constant τ :

$$\frac{1}{\tau_{\text{eff}}} = \frac{1}{\tau} + f.$$

**Bayesian algorithm execution: estimating
computable properties of black-box functions
using mutual information**
W. Neiswanger et al. (2021)

Bayesian optimization can be generalized to black-box algorithms \mathcal{A} estimating some property of a function f from a set of points $(x_1, y_1), \dots, (x_n, y_n)$ (execution

path), with increasing precision as $n \rightarrow \infty$: optimization, numeric integration, root finding. Use the acquisition function

$$\text{EIG}(x) = H[\mathcal{A}|\text{Data}] - \mathbb{E}_{y_x \sim \text{GP}(\text{Data})} H[\mathcal{A}|\text{Data} \cup \{(x, y)\}],$$

approximated using the execution path and/or approximate Bayesian computation (ABC).

**Fast and accurate network embeddings
via very sparse random projection**
H. Chen et al. (2019)

Node embeddings are often computed in two steps:

- Form a node similarity matrix: DeepWalk, *i.e.*, A^k , or, more generally, $\sum_{\ell=1}^k \alpha_\ell A^\ell$;
- Reduce its dimension: SVD, or skipgram, *i.e.*, factorization of

$$M_{uv} = \log \frac{S_{uv} \cdot |S|}{D_u \cdot D_v} - b.$$

Use:

- A sparse random projection

$$P_{ij} = \begin{cases} +\sqrt{s} & \text{with probability } 1/2s \\ 0 & \text{with probability } 1 - 1/s \\ -\sqrt{s} & \text{with probability } 1/2s \end{cases}$$

- Normalize A^k (its entries have a heavy-tailed distribution):

$$\widetilde{A_{ij}^k} = A_{ij}^k \left(\frac{d_j}{2m} \right)^{\lambda-1}$$

where the normalization strength $\lambda - 1$ is chosen by the user.

Available in Neo4j.

**Database-friendly random projections:
Johnson-Lindenstrauss with binary coins**
D. Achlioptas (2002)

**Accurate intelligible models with pairwise
interactions**
Y. Lou et al. (2013)

Generalized additive models with interactions (GA²M), aka *explainable boosting machines* (EBM) can be estimated efficiently (greedily, after estimating and fixing the 1-dimensional functions, and approximating the interactions on the residuals using cuts).

Implementation in `interpret` (R/Python).

**Landscape of R packages for explainable
artificial intelligence**
S. Maksymiuk et al. (2021)

Start with DALEX (also available in Python), `flashlight` or `iml`; look at: feature importance (permutations), Shapley score, ceteris paribus curve, partial dependence plot.

(In Python: aix360, eli5, interpret, lime, shap, skater.)

AutoGL:
a library for automated graph learning
C. Guan et al.

Uncertainty, volatility and persistence norms of financial time series
S. Rudkin et al. (2021)

Multilevel Monte Carlo methods
M.B. Giles

Monte Carlo variance reduction with control variates uses

$$E[f] = E[f - \lambda(g - E[g])]$$

where $E[g]$ is known; the optimal value of λ is

$$\lambda = \text{Cor}(f, g) \sqrt{\frac{\text{Var } f}{\text{Var } g}}.$$

Two-level MLMC uses $E[P_1] = E[P_0] - E[P_1 - P_0]$ where P_0 is a cheap approximation of P_1 (and we have taken $\lambda = 1$).

More generally, MLMC uses

$$E[P_L] = E[P_0] + \sum E[P_\ell - P_{\ell-1}]$$

where the expectation at level ℓ uses $N_\ell \propto \sqrt{V_\ell/C_\ell}$ samples, and V_ℓ , C_ℓ are the variance and cost of one sample.

Generalizations include:

- Randomized MLMC: use a fixed number N of samples, but randomly select the level ℓ of each;
- Richardson approximation;
- Multi-index monte Carlo (time and space discretizations);
- Non-geometric grid;
- Quasi Monte Carlo.

Applications include: SDEs (Euler and Milstein), Lévy processes, SPDE, nested simulations

$$E_Z[f(E_W[g(Z, W)])].$$

Parametric integration is related: to estimate $E[f_\lambda(X)]$ as a function of λ , first estimate $E[f_0(X)]$, $E[f_1(X)]$, then $E[f_{1/2}(X)]$ using $(f_0 + f_1)/2$ as a covariate and continue recursively.

Multilevel Monte Carlo path simulation
M.B. Giles

Pair-copula constructions for financial applications: a review
K. Aas (2016)

Financial applications of vine copulas in finance include:

- VaR or CVaR for small portfolios;
- Vine CAPM (with C-vines (stars) or R-vines (unconstrained));
- Credit risk (probability of default), systemic risk (CDS), liquidity risk (bid-ask spread), operational risk;
- Scenario generation, for CVaR optimization or multivariate option pricing.

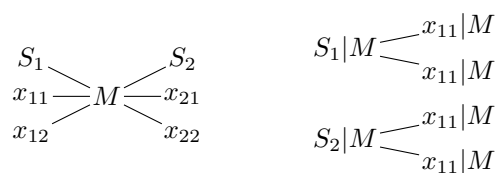
Bayesian inference for latent factor copulas and application to financial risk forecasting
B. Schamberger et al. (2017)

In the 1-factor latent model, $u_i \perp\!\!\!\perp u_j | v$, we can infer the values of v (unobserved) and of the copulas c_j of (u_j, v) (assuming their families, Gaussian or Gumbel, are known, and parametrizing them with the Fisher z transform of their Kendall τ 's, $z = \tanh^{-1}(\tau)$) with Gibbs sampling and *adaptive rejection metropolis sampling* (ARMS).

Application: portfolio VaR.

Asymmetric CAPM dependence for large dimensions: the canonical vine autoregressive model
A. Heinen and A. Valdesogo

As an alternative to the multivariate GARCH model, for more than 30 assets, use a *C*-vine (star-shaped), whose bivariate copula parameter vary with time in a DCC fashion, with GARCH residuals

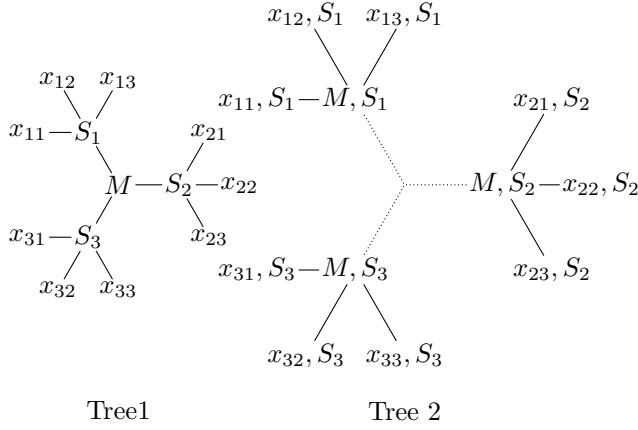


and a Gaussian copula for $x_{ij}|M, S_1, S_2$ where M is the market, S_i the sectors, x_{ij} the stocks in sector i .

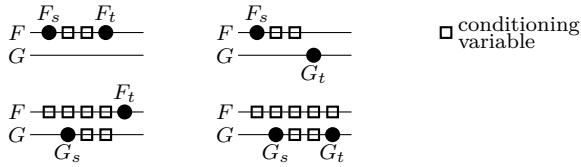
Risk management with high-dimensional vine copulas: an analysis of the Euro Stoxx 50
E.C. Brechmann and C. Czado (2013)

A regular (*i.e.*, unconstrained) vine copula (with constant parameters) with GARCH marginals, estimated on 50 stocks and indices (Euro Stoxx 50 and country indices) performs better than CAVA, a canonical (*i.e.*, star-shaped) vine, with time-varying parameters and

GARCH marginals.



**Copula-based factor models
for multivariate asset returns**
E. Ivanov et al. (2017)



**COPAR: multivariate time series modeling
using copulas autoregressive model**
E.C. Brechmann and C. Czado (2012)

A univariate copula-AR(p) model specifies copulas for

$$\begin{aligned} &X_{n-1}, X_n \\ &X_{n-2}, X_n | X_{n-1} \\ &X_{n-3}, X_n | X_{n-1}, X_{n-2} \\ &\vdots \\ &X_{n-p}, X_n | X_{n-1}, \dots, X_{n-p+1}. \end{aligned}$$

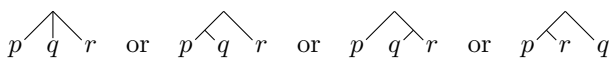
The COPAR model similarly generalizes the VAR(p) model; for two variables and $s < t$:

$$\begin{aligned} &X_s, X_t | X_{s+1:t-1} \\ &X_s, Y_t | X_{s+1:t} \\ &Y_s, X_t | X_{1:t-1}, Y_{s+1:t-1} \\ &Y_s, Y_t | X_{1:t}, Y_{s+1:t-1}. \end{aligned}$$

**The realized hierarchical archimedean copula
in risk modelling**

O. Okhriv and A. Tetereva

Estimate the structure of a *hierarchical archimedean copula* (HAC – an alternative to the pair copula construction (PCC) of vine copulas) by testing if each triplet of variables p, q, r is closer to



from the (rank) correlation (of intraday returns) – contrary to a dendrogram, the tree is not constrained to be binary. Estimate the copula parameters (the copula family is assumed known) by “inverting” the average correlation (there can be more than two variables at a time).

**Nonparametric estimation of the tree structure
of a nested Archimedean copula**
J. Segers and N. Uyttendaele (2018)

A *rooted tree* on a finite set of nodes (leaves) X is a set of subsets $\lambda \in \mathcal{P}(X)$ such that

- (i) $X \in \lambda$;
- (ii) $\forall x \in X \quad \{x\} \in \lambda$;
- (iii) $\forall A, B \in \lambda \quad A \subset B$ or $B \subset A$ or $A \cap B = \emptyset$.

The triplet trees can be estimated from data (for a known, 1-parameter copula family) with statistical tests on Kendall’s τ . If they are not compatible, lower the confidence level of the tests, α , until they are.

Implicit generative copulas
T. Janke et al.

To fit a nonparametric copula to data, train a neural network with two components:

- The first transforms $N(0, 1)$ random data into another distribution Q , without paying attention to the margins;
- The second makes the margins of Q uniform, to have a copula, not with the (sample) probability integral transform (it is not differentiable), but with a soft-rank,

$$\begin{aligned} v_i &= \frac{1}{M} \frac{1}{2} \sum_{j=1}^M \frac{1}{1 + \exp \alpha(y_i - y_j)} \\ &\approx \frac{1}{M} \sum_j \mathbf{1}_{y_i \leq y_j}. \end{aligned}$$

Use the *energy distance* as loss function

$$d^2 = \mathbb{E}_{\substack{U, U' \sim \text{data} \\ V, V' \sim \text{network}}} [2 \|U - V\| - \|U - U'\| - \|V - V'\|].$$

**Copulas as high-dimensional generative
models: vine copula autoencoders**
N. Tagasovska et al.

Learn a low-dimensional representation of the data with an autoencoder; fit a vine copula to that representation; use it to sample new data.

Principal component-guided sparse regression
J.K. Tay et al (2018)

Ridge regression (and elastic net) slightly bias the solution towards the leading singular vectors of X . **PClasso** provides a more aggressive shrinkage by replacing the ridge penalty $\beta' \beta$ with $\beta' V Z V' \beta$ where $X = U D V'$ is the SVD and $Z = \text{diag}(d_1^2 - d_k^2)_{1 \leq k \leq m}$.

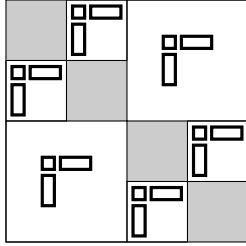
If the factors are known to form groups, treat each separately.

The contours of the plasso penalty are ℓ^1 balls more or less elongated in the direction of the first singular vectors.

***Efficient scalable algorithms
for hierarchically semiseparable matrices***

S. Wang et al.

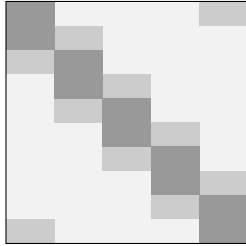
HSS matrices are block matrices, with dense diagonal blocks and low-rank off-diagonal blocks.



The fast kernel transform

J.P. Ryan et al.

Kernel methods require matrix-vector products $x \mapsto Kx$, where the kernel K may be large. To speed up computations, one can use a low-rank approximation of K (the Nyström method uses a random sample of the columns). Fast multipole methods use a low-rank approximation of the off-diagonal blocks, but they require a kernel-specific (Laurent series) expansion of the kernel.



Modern automatic differentiation (AD) tools can automatically compute those expansions.

Implementation in `fastKernelTransform.jl`.

A short course on fast multipole methods

R. Beatson and L. Greengard

We want to compute $u(x_i) = \sum_j w_j k(x_i, x_j)$ for some specific kernel k (it is a matrix-vector product $u = kw$) using a low-rank approximation

$$k(x, y) = \sum_{k=1}^p \phi_k(x) \psi_k(y)$$

(at least for off-diagonal blocks of k).

For instance, in dimension 1, for

$$u(x) = \sum_j w_j \sqrt{(x - x_j)^2 + c^2},$$

one can use a Laurent expansion in x around (away from) x_j ; it can be used with a hierarchical decomposition of \mathbf{R} into “panels” T : use the exact function

$$s_T(x) = \sum_{j \in T} w_j \sqrt{(x - x_j)^2 + c^2}$$

for the panel containing x and nearby panels, and the Laurent expansion for distant ones.

For the *Gauss transform* in dimension 2,

$$U(x) = \sum_{i=1}^N w_i e^{-\|x - x_i\|^2}$$

use the expansion into Hermite functions

$$e^{-\|x - x_i\|^2} = \sum_{n_1, n_2 \geq 0} \Phi_{n_1, n_2}(x - c) \Psi_{n_1, n_2}(x_i - c)$$

$$\Phi_{n_1, n_2}(\mathbf{x}) = h_{n_1}(x) h_{n_2}(y)$$

$$\Psi_{n_1, n_2}(\mathbf{x}) = \frac{x^{n_1} y^{n_2}}{n_1! n_2!}$$

The *multipole expansion*, in dimension 2 ($z \in \mathbf{C}$) is

$$\begin{aligned} \phi(z) &= \sum_i q_i \log(z - z_i) \\ &= \left(\sum_i q_i \right) \log z + \sum_{k \geq 1} \frac{1}{k} \sum_i q_i \frac{z_i^k}{z^k}. \end{aligned}$$

This gives an $O(N \log N)$ algorithm and, with more work (formula to translate a multipole expansion and/or convert it to a Taylor expansion) $O(N)$.

This generalizes to dimension 3 (but the expansion is more complicated).

***Do vision transformers see
like convolutional neural networks?***

M. Raghu et al.

The *centered kernel alignment* (CKA) compares neural network layers (activations) $X \in \mathbf{R}^{m \times p_1}$, $Y \in \mathbf{R}^{m \times p_2}$ on m samples (with p_1 and p_2 neurons) using the Gram matrices $K = XX'$, $L = YY'$.

$$\text{CKA}(K, L) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K) \text{HSIC}(L, L)}}$$

$$H = I_n - \frac{1}{n} \mathbf{1} \mathbf{1}'$$

$$K_c = H K H$$

$$L_c = H L H$$

$$\text{HSIC}(K, L) = \frac{\text{vec}(K_c) \cdot \text{vec}(L_c)}{(m-1)^2}$$

Text generation with efficient (soft) Q-learning

H. Guo et al.

Policy-gradient is on-policy: it is not data-efficient. Q -learning is off-policy, but noisy, and it converges slowly. Try the following variants (soft Q -learning):

- Q -learning with an entropy penalty;
- In the Bellman equation, replace $\text{Max}_a Q(s_{t+1}, a)$ with $\log \sum_a \exp Q(s_{t+1}, a)$;
- Instead of the state action value Q , path consistency learning (PCL) uses the state value function V and $V(s_t) - \gamma V(s_{t+1}) = r_t - \log \pi(a_t|s_t)$.

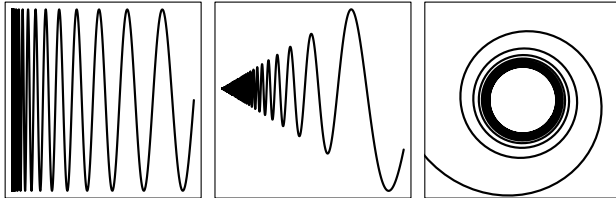
Curiosities and counterexamples in smooth convex optimization

J. Bolte and E. Pauwels (2020)

Given an increasing sequence of compact convex subsets of \mathbf{R}^2 with positively curved boundaries $(C_n)_{n \in \mathbf{Z}}$, there is a \mathcal{C}^k convex function with the C_n 's as level sets, and a positive definite Hessian outside $\text{Argmin } f = \bigcap_{n \in \mathbf{Z}} C_n$. The proof uses:

- The parametrization of ∂C_n by its normals, $c_n : \mathbf{R}/2\pi\mathbf{Z} \rightarrow \partial C_n$
- The boundary of the Minkowski sum $\lambda C_n + (1 - \lambda)C_{n+1}$: $\lambda c_n + (1 - \lambda)c_{n+1}$;
- Bernstein interpolation to smooth piecewise linear functions.

This can be used to build counter-examples, corresponding to paths of infinite length, jiggling or spiralling (e.g., the Newton flow need not converge, the Tikhonov path may have infinite length, etc.)



VICReg: variance-invariance-covariance regularization for self-supervised learning

A. Bardes et al.

Siamese networks learn image representations by maximizing the agreement between embeddings of different views of the same image. To avoid collapse problems, use a loss made of three terms:

- Invariance: $s(z, z') = \frac{1}{n} \sum \|z_i - z'_i\|_2^2$;
- Variance: $\frac{1}{d} \sum \text{hinge}_\gamma \sqrt{\text{Var } z_j}$;
- Covariance: $c(z) = \frac{1}{d} \sum_{j_1 \neq j_2} \text{Cov}(z)_{j_1, j_2}^2$.

Compositional processing emerges in neural networks solving math problems

J. Russin et al.

Tensor product (TP) transformers replace the atten-

tion mechanism

$$\text{Att}(Q, K, V) = \text{softmax} \left(\frac{QK'}{\sqrt{d}} \right) V$$

with

$$\text{TP-Att}(Q, K, V, R) = \text{softmax} \left(\frac{QK'}{\sqrt{d}} \right) V \odot R.$$

The devil is in the detail: simple tricks improve systematic generalization of transformers

R. Csordás et al.

Tests on human language to code translation datasets suggest the following advice.

- Use *universal transformers*, i.e., transformers with weights shared between layers: they are less sensitive to the order of the operations seen during training.
- Prefer relative (vs absolute) positional embedding, to help generalization to longer output lengths.
- Avoid early stopping for model selection.
- Have validation and test sets for both the iid and generalization splits.
- Use accuracy (not loss) for model selection.
- Try different scaling: $N(0, 1)$, Glorot multiplied by \sqrt{d} , Kaiming divided by \sqrt{d} .

Optimal design generation and power evaluation in T: the skpr package

T. Morgan-Wall and G. Khoury (2021)

There are many notions of “optimal design” (choice of a binary model matrix X):

- D-optimal: maximize $\det(X'X)$
- I-optimal: minimize $\text{tr}[(X'X)^{-1}M]$
- A-optimal: minimize $\text{tr}(X'X)^{-1}$
- G-optimal: minimize $\text{tr}(X'X)^{-1}$
- E-optimal: maximize $\sigma_{\min}(X'X)$
- T-optimal: maximize $\text{tr}(X'X)$
- Alias-optimal: minimize $\text{tr}(A'A)$ where $A = (X'X)^{-1}X'X_2$ where X_2 is the model matrix for the interaction terms.

In R: `skpr`, `AlgDesign`, `DoEbase`, `FrF2`, `conf.design`, `planor`, `rsm`.

svars: an R package for data-driven identification in multivariate time series analysis

A. Lange et al. (2021)

VAR models are of the form

$$y_t = \mu + A_1 y_{t-1} + \dots + A_p y_{t-p} + u_t \\ u_t \sim N(0, \Sigma).$$

Structural VAR (SVAR) models further decompose the innovations u_t as

$$u_t = B\varepsilon_t \\ \varepsilon_t \sim N(0, 1)$$

where B is invertible and $\Sigma = BB'$; it is not identifiable: B can be replaced by BO , for $O \in O_n$. To identify it, make additional assumptions, such as:

- Assume Σ changes with time, e.g., $\Sigma_1 = BB'$ until time t_0 , and $\Sigma_2 = B\Lambda B'$, Λ diagonal, after;
- Assume Σ smoothly transitions from Σ_1 to Σ_2 ,

$$\Sigma_t = (1 - \lambda_t)\Sigma_1 + \lambda_t\Sigma_2$$

with a logistic transition function

$$\lambda_t = \frac{1}{1 + e^{-\gamma(s_t - c)}}$$

for some state s_t ;

- Assume $\Sigma_t = B \text{diag}(\text{GARCH}_t) B'$;
- Use independent component analysis (ICA) to find $O \in O_n$, parametrized as a product of $n(n-1)/2$ Givens rotations, using:
 - The Cramer-von Mises distance between the empirical and the independence copulas;
 - FastICA, steadyICA;
 - or a Student likelihood.

***DC3: a learning method
for optimization with hard constraints***
P.L. Donti et al. (2021)

To solve (constrained) optimization problems with deep learning:

- Train a neural net, on solved problems, to produce partial solutions; item Complete them so satisfy the equality constraints (this is differentiable, thanks to the implicit function theorem);
- Perform a few gradient steps, along the manifold defined by the equality constraints, to enforce the inequality constraints (still differentiable).

***A unifying modeling abstraction
for infinite-dimensional optimization***
J.L. Pulsipher et al.

InfiniteOpt.jl solves infinite dimensional optimization problems, *i.e.*, problems involving functions, probability distributions, differential operators, expectations; it automatically discretizes the problem.

A text-based analysis of corporate innovation
G. Bellstam et al. (2017)

Fit a latent Dirichlet allocation (LDA) model, with 15 topics, to 800,000 analyst reports for S&P companies; identify the “innovation” topic as the closest (for the KL divergence) to the word distribution of a textbook on innovation (first Google hit for “innovation textbook pdf”); combine with the Loughran-McDonald sentiment lexicon.

***Language and domain specificity:
a Chinese financial sentiment dictionary***
Z.Du et al. (2021)

Using news articles from finance.sina.com.cn and manually selected words in 3 sentiment categories (positive, negative, and a China-specific “politically positive”, from 500 articles), train a word2vec model to find similar words and extend the lexicon.

Factor models for Chinese A-shares
M.X. Hanauer et al. (2021)

For China, instead of the Fama-French factors, use market, size, E/P (not B/P), and perhaps “abnormal turnover”.

$$\begin{aligned} \text{Turnover} &= \frac{\text{shares traded}}{\text{shares outstanding}} \\ \text{Abnormal turnover} &= \frac{\text{MA}(\text{turnover}, 20 \text{ days})}{\text{MA}(\text{turnover}, 250 \text{ days})} \end{aligned}$$

A comparison of global factor models
M.X. Hanauer (2019)

Replace the Fama-French factors with market, size, E/P, cash-based profitability, change in assets, momentum.

Spectral factor models
F.M. Bandi (2020)

In the 1-factor model (CAPM), $y_t = \alpha + \beta x_t + \varepsilon_t$, decompose the factor returns $(x_t)_t$ into high and low frequency components (Wold decomposition)

$$y_t = \alpha + \beta^{\text{LF}} x_t^{\text{LF}} + \beta^{\text{HF}} x_t^{\text{HF}} + \varepsilon_t$$

The classical model imposes $\beta^{\text{LF}} = \beta^{\text{HF}}$. Those spectral betas can be obtained from the high and low frequency components of y :

$$\begin{aligned} y_t^{\text{LF}} &= \alpha + \beta^{\text{LF}} x_t^{\text{LF}} + \varepsilon_t^{\text{LF}} \\ y_t^{\text{HF}} &= 0 + \beta^{\text{HF}} x_t^{\text{HF}} + \varepsilon_t^{\text{HF}}. \end{aligned}$$

***Weak supervision and Black-Litterman
for automated ESG portfolio construction***
A. Sokolov et al. (2021)

Train a BERT classifier (on New York Times news – they are already tagged) to recognize ESG articles, and use it as a Black-Litterman view to tilt the portfolio away from companies with a lot of ESG press coverage (alternatively, use this ESG score to adjust your model’s alpha).

***A reinforcement learning approach
to optimal execution***
C.C. Moallemi and M. Wang (2021)

To decide when to trade (a single share), predict future prices with an RNN and trade, at time t , if the predicted price change on $[t, T]$ is negative – otherwise, wait.

The loss $\frac{1}{T} \sum_{i=1}^T (r_i - \text{RNN}_i(x_0))^2$ can be replaced with

$$(r_1 - \text{RNN}_1(x_0))^2 + \sum_{i=2}^T (\text{RNN}_{i-1}(x_1) - \text{RNN}_i(x_0))^2$$

or

$$\sum_{i=1}^m (r_i - \text{RNN}_i(x_0))^2 + \sum_{i=m}^T (\text{RNN}_{i-m}(x_m) - \text{RNN}_i(x_0))^2$$

(temporal difference supervised learning).

Instead of the (unconditional) predicted price change, reinforcement learning uses the *continuation value*

$$C_{[t,T]}(x) = \sup_{\pi} \mathbb{E}_{\pi} \left[\sum_{i=t}^{\tau_{\pi}} x_i = x \right],$$

i.e., it takes early stopping into account.

For the state x , use:

- Time of day;
- Spread, normalized by volatility, or price;
- Depth (level1, level5, imbalance $(1 - 5)/(1 + 5)$);
- Number of trades, number of price changes;
- Trade size (exponentially weighted).

Normalize the prices (and the price changes) by the spread.

The reinforcement learning approach is slightly better, but not much, and it is more data-hungry.

**Portfolio selection:
a target-distribution approach
N. Lassance and F. Vrms (2021)**

Replace the portfolio optimization problem

$$\begin{array}{ll} \text{Find} & w \\ \text{To maximize} & w' \mathbb{E}[X] - \frac{1}{2} \lambda w' \text{Var}[X] w \end{array}$$

with

$$\begin{array}{ll} \text{Find} & w \\ \text{To minimize} & \text{KL}(w'X \parallel \text{target}) \end{array}$$

where the target distribution has the same first two moments as an efficient portfolio, but with skewness and kurtosis reflecting the investor's preferences, e.g., a generalized Gaussian

$$f(x) \propto \exp -\frac{1}{2} \left(\frac{|x - \alpha|}{\beta} \right)^{\gamma}$$

or a shifted normal.

If the target distribution has a smaller support, use the “conditionnal KL divergence”

$$\text{CKL}(f \parallel g) = \mathbb{E}_{X \sim f} \left[\log \frac{f(X)}{g(X)} \mid X \in \text{Supp } g \right] + \frac{p}{1-p}$$

where $p = \mathbb{P}_{X \sim f}[X \notin \text{Supp } g]$.

When the asset returns are Gaussian (and the target mean and variances on or above the efficient frontier), then the optimal portfolio is mean efficient.

**Hidden errors in regression-based attribution
L. Sneddon**

If the return forecasts are a linear combination of “alphas”, $\mu = \sum \alpha_i$, the (unconstrained) optimal portfolio is

$$\begin{aligned} w &= \underset{w}{\text{Argmax}} w' \mu - \frac{1}{2} \lambda w' V w \\ &= (\lambda V)^{-1} \mu \\ &= \sum (\lambda V)^{-1} \alpha_i \end{aligned}$$

which gives an exact decomposition of the returns

$$r'w = \sum r'(\lambda V)^{-1} \alpha_i.$$

**Estimation of a high-dimensional
counting process without penalty
for high-frequency events
L. Mucciante and A. Sancetta (2021)**

The explanatory variances X_t for the intensity λ_t of a counting process N_t (here, high-frequency trades) often measure some form of “activity”: they are positive and have a positive impact on the number of events – use a sign constraint in the model.

$$\begin{array}{ll} \text{Find} & \beta \\ \text{To minimize} & -2 \int_0^T X'_t \beta dN_t + \int_0^T (X'_t \beta)^2 dt \\ \text{Such that} & \beta \geq 0 \\ \text{Where} & dN_t = \lambda_t dt \\ & \lambda_t = X'_t \beta \\ & X_t \geq 0 \\ & \lambda_t > 0 \end{array}$$

Predictors include:

- Volume imbalance, for levels 1 to 5,

$$\frac{\text{bid size} - \text{ask size}}{\text{bid size} + \text{ask size}};$$

- Trade imbalance;
- Spread;
- Duration.

and their exponentially weighted moving average.

**The calculus of M-estimation in R with geex
B.C. Saul and M.G. Hudgens (2020)**

Estimators defined by *estimating equations*

$$\sum \psi(y_i, \hat{\theta}) = 0$$

are asymptotically normal and their asymptotic variance can be computed with the sandwich estimator.

The **geex** package computes user-defined estimators and their asymptotic variance (with numerical derivatives, not automatic differentiation).

The calculus of M-estimation
L.A. Stefanski and D.D. Boos (2001)

M-estimators (or generalized estimating equations, GEE) are solutions of $\sum_i \psi(y_i, \theta) = 0$ or, more generally, $\sum_i \psi(y_i, \theta) = c_i$ with $c_n/\sqrt{n} \xrightarrow{p} 0$.

Examples include:

- Mean: $\psi(y, \theta) = y - \theta$;
- Mean and MAD:

$$\psi(y, \theta) = \begin{pmatrix} |y - \theta_2| - \theta_1 \\ y - \theta_2 \end{pmatrix};$$

- Any maximum likelihood estimator (MLE), with the score function

$$\psi(y, \theta) = \frac{\partial \log f_\theta(y)}{\partial \theta};$$

- Mean and variance:

$$\psi(y, \theta) = \begin{pmatrix} y - \theta_1 \\ (y - \theta_1)^2 - \theta_2 \end{pmatrix}$$

(it is not the same ψ function as for the MLE, but it gives the same estimator);

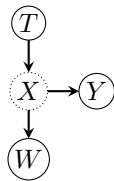
- Ratio \bar{Y}/\bar{X} :

$$\psi(y, \theta) = y - \theta x \quad \text{or} \quad \psi(y, \theta) = \begin{pmatrix} y - \theta_1 \\ x - \theta_2 \\ \theta_1 - \theta_3 \theta_2 \end{pmatrix};$$

- $\mu, \sigma^2, \sigma, \log \sigma$:

$$\psi(y, \theta) = \begin{pmatrix} y - \theta_1 \\ (y - \theta_1)^2 - \theta_2 \\ \sqrt{\theta_2} - \theta_3 \\ \log \theta_2 - \theta_4 \end{pmatrix};$$

- Instrumental variables $\beta_{IV} = \beta_{Y|T}/\beta_{W|T}$:



$$\psi(y, \theta) = \begin{pmatrix} \theta_1 - t \\ (y - \theta_2 w)(\theta_1 - t) \end{pmatrix}$$

$$\text{or } \psi(y, \theta) = \begin{pmatrix} \theta_1 - t \\ \theta_2 - w \\ (y - \theta_3 w)(\theta_2 - w) \\ (y - \theta_4 w)(\theta_1 - t) \end{pmatrix};$$

- Robust location $\psi(y, \theta) = h_k(y - \theta)$ where $h_k = \mathbf{V}$;
- Sample quantile: $\psi(y, \theta) = p - \mathbf{1}_{y \leq \theta}$
- Positive mean deviation from the median:

$$\psi(y, \theta) = \begin{pmatrix} 2(y - \theta_2)\mathbf{1}_{y > \theta_2} - \theta_1 \\ \frac{1}{2} - \mathbf{1}_{y \leq \theta_2} \end{pmatrix};$$

- Non-linear least squares, $y = g(x, \beta) + \varepsilon$:

$$\psi(y, x, \theta) = [y - g(x, \beta)]g'(x, \beta);$$

- Linear regression with robust loss (truncated L^1):

$$\psi(y, x, \theta) = h_k(y - x'\beta)x;$$

- Generalized linear models (GLM).

Under reasonable assumptions,

$$\hat{\theta} \sim \text{AMN}\left(\theta_0, \frac{V(\theta_0)}{n}\right)$$

(asymptotically multivariate normal), where

θ_0 : true parameter (unknown)

$\hat{\theta}$: estimator

$$A(\theta_0) = \mathbb{E}_{Y \sim F} \left[- \frac{\partial \psi(Y, \theta)}{\partial \theta} \Big|_{\theta=\theta_0} \right]$$

$$B(\theta_0) = \mathbb{E}_{Y \sim F} [\psi(Y, \theta_0) \psi(Y, \theta_0)']$$

$$V(\theta_0) = A(\theta_0)^{-1} B(\theta_0) A(\theta_0)^{-\top} \quad (\text{sandwich estimator})$$

For the MLE, $A = B = I$ is the information matrix and $V = I^{-1}$, but the general formula remains valid even under model misspecification. Use a CAS to compute the matrix products: often, you only want one of the diagonal elements of V (your estimator alone is not an M-estimator, but it is a component of one).

The variance matrix can also be used for Wald tests: $(\hat{\theta} - \theta_0)' V_n(\hat{\theta})^{-1} (\hat{\theta} - \theta_0)$.

For non-smooth functions, swap ∂_θ and \mathbb{E} :

$$A(\theta_0) = - \frac{\partial}{\partial \theta} \mathbb{E}_{Y \sim F} [\psi(Y, \theta)] \Big|_{\theta=\theta_0}.$$

An axiomatization of Λ -quantiles
F. Bellini and I. Peri (2021)

The Λ -quantiles of a distribution F are

$$\Lambda\text{-VaR}(R) = -\inf \{x \in \mathbf{R} : F(x) > \Lambda(x)\}$$

where $\Lambda : \mathbf{R} \rightarrow (0, 1)$ is non-decreasing; they are characterized by the *locality* property: changing F above or below $\Lambda\text{-VaR}(F)$ does not change $\Lambda\text{-VaR}(F)$.

Nonparametric extrema analysis
in time series for envelope extraction,
peak detection and clustering
K. Gokcesu and H. Gokcesu

To detect peaks (or bursts) in a signal $(x_t)_t$, compute its envelope: signals a, b , such that $\forall t \ a_t \leq x_t \leq b(t)$ and $\forall t \ x_t \in \{a_t, b_t\}$, minimizing

$$\sum |a_{t+1} - a_t| + \sum |b_{t+1} - b_t|.$$

If a and b are defined using constant interpolation,

$$a_t = s_t x_t + (1 - s_t) a_{t-1}$$

$$b_t = (1 - s_t) x_t + s_t b_{t-1}$$

$$s_t \in \{0, 1\}.$$

The problem can be solved efficiently, in $O(T \log T)$.

Iterated and exponentially weighted moving principal component analysis
P. Bilokon and D. Finkelstein (2021)

When computing principal components on an expanding or moving window, the principal components can flip sign: instead of recomputing them anew each time, use an iterative algorithm (Ogita-Aishima) to refine the previous ones.

To account for non-stationarity, use exponential weights.

(Python implementation available.)

Iterative refinement for symmetric eigenvalue decomposition
T. Ogita and K. Aishima (2018)

Principal component analysis: a review and recent developments
I.T. Jolliffe and J. Cadima (2016)

PCA can be computed from the eigendecomposition of the variance matrix $S = \text{Var } X$: the unit eigenvectors a satisfy $\text{Var } Xa = a'Sa = \lambda aa' = \lambda$, or from the SVD of the column-centered data matrix $X^* = ULA'$,

$$(n-1)S = X^*X^* = (ULA')'(ULA') = AL^2A',$$

which also gives the *biplot*: the rows of U represent the observations, those of AL the variables.

Variants include:

- Varimax: rotate the principal components to make the loadings more concentrated;
- Sparse PCA (SCoTLASS);
- Robust PCA,

$$\text{Minimize } \|L\|_* + \lambda \|S\|_1 \text{ such that } X = L + S;$$

- Functional PCA;
- Symbolic PCA (PCA for intervals, histograms, etc.).

Evaluation of the importance of criteria for the selection of cryptocurrencies
N.A. van Heerden et al.

To combine predictors x_1, \dots, x_m of y , use a linear combination with weights:

$$\begin{aligned} w_j &= 1/m \\ w_j &\propto \sigma_j \\ w_j &\propto \sigma_j \sum_i (1 - \rho_{ij}) \end{aligned}$$

$$w_j \propto H_j \text{ (entropy)}$$

$$w_j \propto \frac{d_j^-}{d_j^- + d_j^+}$$

$$\begin{aligned} \text{where } d_j^+ &= d(x_j, y) \\ d_j^- &= d(x_j, -y). \end{aligned}$$

Learning-based robust optimization: procedures and statistical guarantees
L.J. Hong et al. (2017)

The chance-constrained program

$$\begin{aligned} &\text{Find } x \\ &\text{To minimize } f(x) \\ &\text{Such that } \mathbb{P}_{\xi \sim P}[f(x, \xi) \in A] \geq 1 - \varepsilon \end{aligned}$$

can be relaxed to a robust optimization problem by replacing the constraint with

$$\forall \xi \in U \ g(x, \xi) \in A$$

for some set U (uncertainty set) such that

$$\mathbb{P}_{\xi \in P}[\xi \in U] \geq 1 - \varepsilon.$$

If P is only known from a sample, split the data in two:

- Use the first part to define the shape of U (ellipsoid, convex hull, union of ellipsoids or polytopes);
- Use the second to calibrate the size of U , such that

$$\mathbb{P}_{\text{data} \sim P^n} \left[\mathbb{P}_{\xi \sim P}[\xi \in U(\text{data})] \geq 1 - \varepsilon \right] \geq 1 - \delta.$$

From data to decisions: distributionally robust optimization is optimal
B.P.G. van Parys et al.

For the optimization problem

$$\begin{aligned} &\text{Find } x \\ &\text{To minimize } c(x, P) \\ &\text{Where } c(x, P) = \mathbb{E}_{\xi \sim P} \gamma(x, \xi) \end{aligned}$$

use the distributionally robust predictor

$$\hat{c}(c, \hat{P}) = \sup_{\text{KL}(\hat{P} \| P) \leq r} c(x, P)$$

where \hat{P} is the empirical distribution: the out-of-sample disappointment $P[c(x, P) > \hat{c}(x, \hat{P})]$ decays exponentially (uniformly) for all x .

Fractional growth portfolio investment
A.E. Brockwell (2021)

The Kelly portfolio, investing fractions $k = \Sigma^{-1}(\mu - r\mathbf{1})$ of the wealth in each asset, maximizes the expected log-return

$$L = \mathbb{E} \left[\delta^{-1} \log \frac{A_{t+\delta}}{A_t} \right],$$

but the variance

$$V = \text{Var} \left[\delta^{-1} \log \frac{A_{t+\delta}}{A_t} \right]$$

can be very high. The *fractional Kelly portfolio* invests a fraction αk , $\alpha \in [0, 1]$, and has log-return expectation and variance

$$\begin{aligned} L &= r + (\alpha + \frac{1}{2}\alpha^2)S^2 \\ V &= \alpha^2 S^2 \end{aligned}$$

where $S = \sqrt{(\mu - r\mathbf{1})'\Sigma^{-1}(\mu - r\mathbf{1})}$ is the Sharpe ratio of the Kelly portfolio.

Arbitrage-free implied volatility surface generation with variational autoencoders
B.X. Ning et al.

To generate arbitrage-free synthetic implied volatility (IV) surfaces:

- Fit a stochastic differential equation (SDE) to the volatility surface, for each day;
- Train a VAR on the SDE parameters;
- Sample from the VAE;
- Compute the IV surface from the corresponding SDE.

Regularization is all you need: simple neural nets can excel on tabular data
A. Kadra et al.

A simple MLP with a cocktail of regularizers, with suitable hyperparameter tuning (BOHB, 4 days), outperforms traditional machine learning (gradient boosted trees) on tabular data:

- Batchnorm;
- Stochastic weight averaging, *i.e.*, averaging the local optima encountered on the optimization path;
- Lookahead optimizer;
- Snapshot ensembles: ensembles of minima obtained after restarts;
- Weight decay (L^2);
- Dropout (various shapes);
- Skip connections (shakedrop, shake-shake);
- Mixup: data augmentation with convex combinations of training samples
- Cutout: mask subsets of the input;
- Cutmix: Mixup + Cutout;
- FGSM (adversarial training).

Examining the dynamic asset market linkages under the covid-19 global pandemic
A. Noda (2021)

To test for market efficiency (between 3 markets, S&P 500, Bitcoin, gold), estimate a VAR

$$x_t = \mu + A_1 x_{t-1} + \dots + A_q x_{t-q} + \varepsilon_t$$

with time-varying coefficients A_i , and compute the “joint degree of market efficiency” ζ

$$\Phi = (I - A_1 - \dots - A_q)^{-1}$$

$$\zeta = \text{Max} \sqrt{(\Phi - I)'(\Phi - I)}.$$

Under the efficient market hypothesis (EMH), $A_1 = \dots = A_q = 0$ and $\zeta = 0$.

Those markets are getting less efficient.

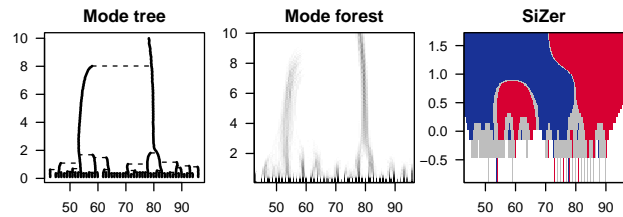
multimode: an R package for mode assessment
J. Ameijeiras-Alonso et al. (2021)

To visually assess the presence of several models in univariate data:

- The *mode tree* plots the modes versus the bandwidth of a kernel density estimator (kde);

- The *mode forest* is a set of bootstrapped mode trees;
- The significant zero (*SiZer*) map plots the significance of the sign of the slope of a kde (whether a confidence interval for $f'_h(x)$ is in \mathbf{R}_- , \mathbf{R}_+ or contains zero) as a function of x and the bandwidth h .

There are also statistical tests.



mosum: a package for moving sums in change-point analysis
A. Meier et al. (2021)

To detect changepoints (on the mean), compute the average on a window of size G before and after all potential breakpoints k

$$T_G(k) = \sqrt{\frac{G}{2}} \left(\text{Mean}_{k < t \leq t+G} X_t - \text{Mean}_{k-G < t \leq t} X_t \right)$$

and return $\text{Argmax}_k |T_G(k)| / \sigma_k$. You can try several (or asymmetric) bandwidths.

Conformal prediction with Orange
T. Hočevar et al. (2021)

A *conformal predictor* predicts a *set* of labels with low nonconformity, for some user-chosen nonconformity function such as:

- Distance to the closest neighbour with the same label;
- Predicted probability of the correct class; s
- Difference of predicted probabilities of the two most likely classes;
- (For regression) absolute error of the prediction.

Also check **conformal** (R), **nonconformist** (Python).

gdpc: an R package for generalized dynamic principal components
D. Peña et al. (2020)

The first *dynamic principal component* of a multivariate time series $(z_t)_t$ is the univariate time series $(f_t)_t$ such that (f_{t-k}, \dots, f_t) gives the best reconstruction of z_t :

$$z_{it} \approx \alpha_i + \sum_{j=0}^k \beta_{ij} f_{t-j}$$

(it is noisy at the ends of the sample and therefore not useful for forecasting); β can be computed from f with OLS and conversely. The loss can be a square loss, or a robust alternative. The original DPC was a linear combination (and convolution) of the observations.

**Dimension reduction for time series
in a blind source separation context using R**
K. Nordhausen et al. (2021)

Diagonalize, not just the covariance matrix, but (jointly) several cross-covariance matrices, for several lags.

Blind source separation of a multivariate time series $(z_t)_t$ assumes that the signals are autocorrelated, but that the noise is not:

$$x_t = \mu + \Omega z_t$$

$$\text{Cov}_\tau(z_t) = \text{E}[z_t z'_{t+\tau}] = \begin{pmatrix} \Lambda_\tau & 0 \\ 0 & 0 \end{pmatrix}.$$

AMUSE uses the eigendecomposition of the autocovariance matrix of the standardized time series $x_t^{\text{st}} = U\Lambda_\tau U'$, $UU' = I$. To allow the eigenvalues to be similar, SOBI approximately jointly diagonalizes several autocovariance matrices, maximizing

$$\sum_{\tau \in \mathcal{T}} \sum_i (u'_i \text{Cov}_\tau(x_t^{\text{st}}) u_i)^2, \quad U \text{ orthogonal.}$$

This can be generalized (gFOBI) to include 4th cross-moments $B_\tau = \text{E}[x_{t+\tau} x'_t x'_{t+\tau}]$ by maximizing

$$\sum_\tau \sum_i (u'_i B(x_t^{\text{st}}) u_i)^2$$

or by using a wider class of cumulants (gJADE).

$$\sum_\tau \sum_i \sum_{jk} (u'_i C_\tau^{jk}(x_t^{\text{st}}) u_i)^2$$

$$C_\tau^{jk} = \text{E}[x_{t+\tau} x'_t e_{jk} x'_{t+\tau}] - \text{Cov}_\tau(x_t)(e_{jk} + e_{kj}) \text{Cov}_\tau(x_t)' - \text{tr}(e_{jk}) I_p$$

$(e_{jk})_{jk}$: standard basis of $\mathbf{R}^{p \times p}$

or by inserting a nonlinear function g in the moment computations ($y \mapsto y^2$ or $y \mapsto \log \cosh y$)

$$\sum_{\tau, i} \text{E}[g(u'_i x_t^{\text{st}}) g(u'_i x_{t+\tau}^{\text{st}})^2] - \text{E}[g(u'_i x_t^{\text{st}})] \text{E}[g(u'_i x_{t+\tau}^{\text{st}})^2]$$

or by combining linear and quadratic parts

$$\lambda \sum_{i, \tau} (u'_i C_\tau u'_i)^2 + (1 - \lambda) \sum_{i, \tau} \text{E}[(u'_i x_t^{\text{st}})^2 (u'_i x_{t+\tau}^{\text{st}})^2].$$

For supervised dimension reduction (TSIR, TSAVE, TSSH), approximately jointly diagonalize

$$G_\tau(x_t^{\text{st}}, y_t) = \text{Cov} \text{E}[z_t | y_{t+\tau}]$$

or $G_\tau(x_t^{\text{st}}, y_t) = \text{E}[I_p - \text{Cov}(z_t | y_{t+\tau})]$

by maximizing

$$\sum_{\tau, i} (u'_i G_\tau u_i)^2.$$

Supervised dimension reduction of X given Y looks for a subspace $\text{Span } B$ such that the distributions

$$Y | X = x \quad \text{and} \quad Y | BX = Bx$$

coincide, by using the first eigenvectors of

$$M = \frac{1}{n} \sum_j \bar{z}_j \bar{z}'_j$$

$$M = \frac{1}{n} \sum_j (I - \text{Var}[Z | Y = j])^2$$

$$M = \sum_i y_i z_i z'_i$$

$$M = \sum \text{res}(y \sim x)_i z_i z'_i$$

(SIR (sliced inverse regression), SAVE (sliced average variance estimates), PHD (principal Hessian directions)) where

- z is the standardization of x ;
- y is discrete, with values $j = 1, \dots, h$ (slice it into equal-sized bins if needed);
- $z_j = \text{E}[Z | Y = j]$.

Implementation in `dr`.

**Principal component analysis
with sparse fused loading**
J. Guo et al. (2010)

Adding an L^1 penalty to PCA does not give enough sparsity: relax the orthogonality constraint by looking for A and B such that $X \approx XBB'$ (projection of X on $\text{Span } B$), $B \approx A$, A orthogonal:

Find A, B
To minimize $\|X - XBA'\|^2 + \lambda_1 \sum_k \|\beta_k\|_1 + \lambda_2 \sum_k \|\beta_k\|_2^2$
Such that $AA' = I$.

You can replace the second penalty with a *fusion penalty* to pull the coefficients together when the correlation is high.

$$\lambda_2 \sum_k \sum_{s < t} |\rho_{st}| |\beta_{sk} - \text{sign}(\rho_{st}) \beta_{tk}|$$

**SQUAREM: an R package
for off-the-shelf acceleration of EM, MM
and other EM-like monotone algorithms**
Y. Du and R. Varadhan (2020)

The square iterative method (SQUAREM) speeds up the convergence of EM-like monotone iterative algorithms

$$\theta_{n+1} = F(\theta_n)$$

$$L(\theta_{n+1}) \geq L(\theta_n)$$

by iterating:

$$\begin{aligned}
\theta_1 &= F(\theta_0) \\
\theta_2 &= F(\theta_1) \\
r &= \theta_1 - \theta_0 \\
v &= (\theta_2 - \theta_1) - r \\
\alpha &= -\|r\| / \|v\| \\
\theta_3 &= \theta_0 - 2\alpha r + \alpha^2 v \\
\text{if } L(\theta_3) &> L(\theta_2) - \eta : \\
\theta_0 &= F(\theta_3) \\
\text{else:} \\
\theta_0 &= \theta_2.
\end{aligned}$$

Examples include:

- Poisson mixtures;
- Ecdf estimation from interval-censored data;
- Admixture (estimating the proportion of each ancestral population in each individual, and the proportion of each allele in each ancestry);
- MM for logistic regression;
- Factor analysis

$$\begin{aligned}
Z &\sim N(0, I) && \text{latent} \\
Y &\sim N(\beta'Z, \Delta) && \text{observed.}
\end{aligned}$$

The R package smicd:
statistical methods for interval-censored data
P. Walter

Fully hyperbolic convolutional neural networks
K. Lensink et al.

Hyperbolic networks

$$Y_{j+1} = 2Y_j - Y_{j-1} + f_\theta(Y_j),$$

i.e., discretizations of $\ddot{y} = f_\theta(y)$ are invertible (we can recover Y_{j-1} from Y_j and Y_{j+1}). Combine them with learned discrete wavelet transforms to coarsen the state and increase the dimension without losing information.

Generative ODE modeling
with known unknowns
O. Linial et al. (2021)

To estimate the parameters θ of an ODE $\dot{z} = f_\theta(z)$ for which only an (unknown) transformation of the state $z = g(z)$ is observed, use an autoencoder

$$(x_t)_t \xrightarrow{\text{encoder}} \theta, z_0 \xrightarrow[\dot{z} = f_\theta(z)]{\text{known ODE}} (z_t)_t \xrightarrow{\text{decoder}} (x_t)_t$$

Exterior point operator splitting
for nonconvex learning
S. Das Gupta et al. (2021)

A closed set is *prox-regular* at a point if projection onto it is single-valued in the neighbourhood of this point.

Examples include sparsity and low rank:

$$\begin{aligned}
N &= \{x \in \mathbf{R}^d : \#x \leq k\} \\
N &= \{X \in \mathbf{R}^{m \times d} : \text{rank } X \leq k\}.
\end{aligned}$$

To solve the optimization problem

$$\begin{aligned}
&\text{Find } x \\
&\text{To minimize } f(x) + \frac{1}{2}\beta \|x\|^2 \\
&\text{Such that } x \in C \cap N
\end{aligned}$$

where f is convex, C is convex, N is non-convex but prox-regular around local minima, replace the constraint with an indicator function

$$\begin{aligned}
&\text{Find } x \\
&\text{To minimize } f(x) + \frac{1}{2}\beta \|x\|^2 + \iota(x) \\
&\text{Where } \iota(x) = \begin{cases} 0 & \text{if } x \in C \cap N \\ \infty & \text{otherwise,} \end{cases}
\end{aligned}$$

regularize it

$$\begin{aligned}
\iota_\mu(x) &= \min_y \iota(y) + \frac{1}{2\mu} \|y - x\|^2 \\
&= \frac{1}{2\mu} d(x, C \cap N)^2
\end{aligned}$$

and progressively decrease μ (e.g., from 1 to 10^{-4}).

To solve each subproblem

$$\text{Minimize}_x f(x) + \underbrace{\iota_\mu(x) + \frac{1}{2}\beta \|x\|^2}_{\iota(x)}$$

use ADMM (Douglas-Rachford)

$$\begin{aligned}
x &\leftarrow \text{prox}_{\gamma f} z \\
y &\leftarrow \text{prox}_{\gamma \iota}(2x - z) \\
z &\leftarrow z + y - x
\end{aligned}$$

where $\text{prox}_{\gamma f} z$ is the solution of a convex optimization problem, often known in closed form, and $\text{prox}_{\gamma \iota}$ can be computed from the projection onto $C \cap N$.

Implementation in NExOS.jl.

High frequency covariance: a Julia package
for estimating covariance matrices
using high-frequency financial data
S. Baumann and M. Klymak (2021)

To estimate volatility in presence of microstructure noise:

- Either use large intervals, so that the microstructure noise disappears;
- Or combine the sample volatility estimators at two different scales.

To estimate the covariance from asynchronous time series:

- Choose a grid of times and compute the correlation using the last available log-prices

$$\text{Cov}(X, Y) = \frac{1}{N} \sum_i (X_{\leq t_i} - \bar{X})(Y_{\leq t_i} - \bar{Y})$$

(but the correlations are biased downwards);

- *Refresh time sampling* (BNHLS) uses an adaptive grid where the next time is the time at which each asset has an updated price; you may want to put assets in blocks of similar trading frequencies;
- Pre-averaging averages returns on a moving window and multiplies the returns of two assets when the windows intersect;
- The spectral local method of moments is another estimator;
- The 2-scale volatility estimator can be used to compute covariances:

$$\text{Cov}(X, Y) = \frac{\text{Var}[\gamma X + (1-\gamma)Y] - \text{Var}[\gamma X - (1-\gamma)Y]}{4\gamma(1-\gamma)}$$

To make sure the covariance matrix is positive semi-definite:

- Linearly interpolate with the identity;
- Or replace eigenvalues below a threshold with the average of the (positive) eigenvalues thus discarded.

For the correlation matrix, iterate:

- Take the nearest positive semi-definite matrix;
- Take the nearest matrix with unit diagonal and off-diagonal elements in $[-1, +1]$.

On covariation estimation for multivariate continuous Itô semimartingales with noise in non-synchronous observation schemes
K. Christensen et al. (2011)

To estimate a covariance matrix from asynchronous data, in presence of microstructure noise, first average the returns, with some kernel function, pooling k consecutive ticks, $k \propto n$, then use the Hayashi-Yoshida estimator:

$$\rho_{\text{HY}} = \frac{\sum_{I_i^X \cap I_j^Y \neq \emptyset} \Delta X_i \Delta Y_j}{\sqrt{\sum (\Delta X_i)^2 \sum (\Delta Y_j)^2}}.$$

Estimating the quadratic covariation matrix from observations: local method of moments and efficiency
M. Bibinget et al. (2014)

Estimator of the variance matrix, from asynchronous data, in presence of microstructure noise, from the Fourier coefficients of the time series on subintervals of $[0, 1]$

$$S_{jkl} = \sum_i (Y_{i\ell} - Y_{i-1,\ell}) \Phi_{jk} \left(\frac{t_{i-1,\ell} + t_{i\ell}}{2} \right)$$

$$\Phi_{jk} = \frac{1}{kh} \int_{j \text{ periods}}^{(k+1)h} \cos\left(\frac{2\pi}{k} (t - jh)\right) dt$$

BayesNetBP: an R package for probabilistic reasoning in Bayesian networks
H. Yu et al. (2020)

To use a conditional Gaussian Bayesian network (CG-BN, *i.e.*, a PGM with a mixture of discrete and continuous nodes) estimated by `bnlearn`, `catnet`, `deal`, `pcalg`, or Hugin (commercial).

R0I: an extensible R optimization infrastructure
S. Theußl et al. (2020)

The R0I package provides a unified interface to most optimization packages (linear, mixed integer, conic, quadratic, nonlinear – there is a long list of supported solvers): specify the objective (linear, mixed integer, quadratic or nonlinear), the constraints (linear, second order, semi-definite, (dual) exponential, (dual) power cone, nonlinear), the bounds (and integrality constraints), and the solver (the package can provide a list of solvers suitable for the problem). Some problems can be automatically reformulated (e.g., binary QP to MILP).

Examples include:

- Best subset regression (linear regression with an ℓ_0 penalty);
- Relative risk regression (logistic regression for the relative risk instead of the odds ratio, *i.e.*, with a log link);
- Convex clustering (sum of norm (SON) clustering):

$$\text{Minimize}_M \sum_i \|M_{i\cdot} - X_{i\cdot}\|_2^2 + \lambda \sum_{i < j} \|M_{i\cdot} - M_{j\cdot}\|_1;$$

- Graphical lasso.

CVXR: an R package for disciplined convex optimization
A. Fu et al. (2020)

Long list of examples of convex optimization problems:

- Huber regression (L^2 for small residuals, L^1 for large ones);
- Quantile regression;
- Elastic net;
- Logistic regression ($z \mapsto \log(1 + e^z)$ is convex);
- Graphical lasso

$$\begin{array}{ll} \text{Find} & S \\ \text{To maximize} & \log \det S - \text{tr}(S\hat{\Sigma}) \\ \text{Such that} & S \succcurlyeq 0 \\ & \sum |S_{ij}| \leq \alpha \end{array}$$

- Saturating hinges (linear splines, constant beyond their boundaries)

$$\begin{array}{ll} \text{Find} & w_0, w \\ \text{To minimize} & \sum_k \ell(y_i, f(x_i)) + \lambda \|w\|_1 \\ \text{Such that} & \sum_{j=1}^k w_j = 0 \end{array}$$

- Log-concave distribution estimation on $[0, k]$.

- Survey calibration (changing the weights as little as possible so that the statistics of the sample match those of the population)
- Nearly isotonic fit: replace the constraints $\beta_{i+1} \geq \beta_i$ with a penalty on $(\beta_i - \beta_{i+1})_+$;
- Nearly convex fit: idem with $(\beta_i - 2\beta_{i+1} + \beta_{i+2})_+$
- Worst case risk of a portfolio with constraints on the asset variance matrix (e.g., known variances and correlation signs)

$$\begin{array}{ll}\text{Find} & \Sigma \\ \text{To maximize} & w' \Sigma w \\ \text{Such that} & \Sigma \succeq 0 \\ & \forall i, j \quad L_{ij} \leq \Sigma_{ij} \leq U_{ij}\end{array}$$

- Catenary (with a ground constraint);
- Portfolio optimization;
- Kelly gambling;
- Channel capacity (maximum mutual information);
- Fastest mixing Markov chain on a known directed graph: the rate of convergence is the second largest eigenvalue of P , i.e., $\sigma_{\max}(P - \frac{1}{n}\mathbf{1}\mathbf{1}')$.

cutpointr: improved estimation and validation of optimal cutpoints in R
C. Thiele and G. Hirschfeld (2021)

For more robust cutpoints for (logistic-regression-like) classification algorithms

$$\text{data} \mapsto \text{score} \mapsto \mathbf{1}_{\text{score} \geq \text{cutpoint}}$$

optimize the Youden index

$$\begin{aligned} J &= \text{sensitivity} + \text{specificity} - 1 \\ \text{sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ \text{specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}} \end{aligned}$$

or some other metric, not on the sample data, but:

- After bootstrap (average the optimal cutpoints);
- Or after smoothing the cutpoint \mapsto metric function (GAM, splines, loess);
- Or using a kde of the conditional distributions

$$\text{score} \mid P \quad \text{and} \quad \text{score} \mid N.$$

Colorspace: a toolbox for manipulating and assessing colors and palettes
A. Zeileis et al. (2020)

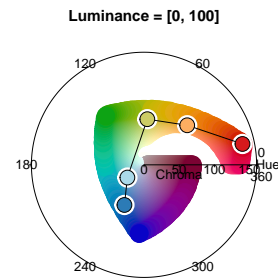
There are many colour spaces:

- RGB;
- sRGB (device-independent: fixed Gamma);
- HSV, whose dimensions are confounded: brightness changes dramatically with hue;
- HLS (very similar);
- XYZ (not perceptually uniform, unintuitive, but standardized);
- LUV: perceptually uniform, for emissive technologies, but the UV dimensions (red/green and yellow/blue) are unintuitive;

- LAB (similar, for pigments);
- polarLUV, aka HCL – that is the one you should use;
- polarLAB.

The package provides many predefined palettes, functions to evaluate a palette (`demoplot`, `hclplot`, `specplot`, `swatchplot`), some interactively (`hcl_wizard`, `hcl_color_picker`, `cvd_emulator`) and also desaturate, lighten, darken.

Many other packages provide colour palettes: `RColorBrewer`, `viridis`, `rcartocolor`, `scico`, `pals`, `paletteer`, `wesanderson`, `Polychrome`; also check `colorscience` (more complex algorithms), `roloc` (English colour names).



FastSHAP: real-time Shapley value estimation
N. Jethani et al.

The Shapley score is the solution of an optimization problem:

$$\begin{array}{ll}\text{Find} & \phi \\ \text{To minimize} & \mathbb{E}_{s \sim p} [v(s) - v(\mathbf{0}) - s' \phi] \\ \text{Such that} & \mathbf{1}' \phi = v(\mathbf{1}) - v(\mathbf{0})\end{array}$$

where p is the Shapley kernel; it can be solved by gradient descent.

If you want Shapley scores for many inputs to the same neural net, learn a function to compute the solution of that problem (for the constraint, either ignore it and modify the solution to satisfy it, or add it as a penalty).

Efficient estimation of bid-ask spreads from open, high, low and close prices
D. Aria et al. (2021)

Assuming that observed log-prices are the latent prices plus or minus half the spread,

$$\begin{aligned} \log P_t &= \log \tilde{P}_t + S \cdot (B_t - \frac{1}{2}) \\ B_t &\sim \text{Bernoulli} \end{aligned}$$

the spread is given by

$$\text{Cov}(\Delta \log P_t, \Delta \log P_{t-1}) = -\frac{1}{4} \text{Spread}^2$$

unless there are periods with no trades. More generally,

$$\text{Spread}^2 = -\frac{4 \text{Cov}(\Delta \log P_t, \Delta \log P_{t-1})}{(1 - P[P_t = P_{t-1}])^2}.$$

This can be extended to OHLC data.

A characterization of cross-impact kernels
M. Rosenbaum and M. Tomas (2021)

Model price impact as

$$P_t - P_0 = \int_0^t K(t-s)(dN_s^a - N_s^b)$$

where (N^a, N^b) is a multi-dimensional Hawkes process with intensity

$$\begin{pmatrix} \lambda_t^a \\ \lambda_t^b \end{pmatrix} = \begin{pmatrix} \mu \\ \mu \end{pmatrix} + \int_0^t \begin{pmatrix} \Phi^{aa}(t-s) & \Phi^{ab}(t-s) \\ \Phi^{ba}(t-s) & \Phi^{bb}(t-s) \end{pmatrix} \begin{pmatrix} dN_s^a \\ dN_s^b \end{pmatrix}$$

and $K : \mathbf{R}_+ \rightarrow \mathbf{R}^{d \times d}$ is the cross-impact kernel.

A martingale-admissible kernel is a kernel for which the prices are a martingale; this is the case if

$$P_t - P_0 = \text{long-term impact} \times \text{long-term order imbalance} \\ = \left(\lim_{s \rightarrow \infty} K(s) \right) \times \left(\lim_{s \rightarrow \infty} \mathbb{E}[N_s^a - N_s^b | \mathcal{F}_t] \right).$$

A no-arbitrage condition imposes constraints on $K(0)$ and $\lim_{s \rightarrow \infty} K(s)$; there is only one martingale-admissible kernel satisfying those (necessary) conditions; it need not ensure the absence of arbitrage, but fits the data well, and one can compute the closest no-arbitrage kernel.

DAGs with NO TEARS:
continuous optimization for structure learning
X. Zheng et al. (2018)

A graph is acyclic if its adjacency matrix satisfies $\text{tr } e^A = d$ – indeed,

$$\text{tr } e^A = \text{tr } I + \text{tr } A + \frac{1}{2!} \text{tr } A^2 + \dots \geq \text{tr } I = d$$

and $\text{tr } A^k$ counts cycles of length k .

For a weight matrix W , use $h(w) = \text{tr } e^{W \odot W} - d$ ($W \odot W$ has the same sparsity as W and nonnegative entries).

Dynotears:
structure learning from time series data
R. Pamfil et al. (2020)

Use the trace exponential function to enforce acyclicity of W in the SVAR model

$$\mathbf{x}_t = W\mathbf{x}_t + A_1\mathbf{x}_{t-1} + \dots + A_p\mathbf{x}_{t-p} + \varepsilon_t.$$

Beyond prediction in neural ODEs:
identification and interventions
H. Aliee et al.

Estimating an ODE from a single trajectory is an ill-posed problem: even in the linear case, the ODE is not uniquely determined:

$$(\forall t \ e^{tA} = e^{tB}) \not\Rightarrow A = B.$$

Adding a regularizer does not help but, in practice, this does not seem to matter, and a simple regularization

$$\|\theta\|_{\text{simple}} = \|W^{L+1}W^L \dots W^1\|_{1,1}$$

(the L^1 norm of the gradient of the neural network, if we remove the nonlinearities) correctly recovers causal relations.

Forecasting stock returns
with large dimensional factor models
A. Giovannelli and D. Massacci (2021)

Statistical factor models explain current stock returns from risk factors computed from current stock returns. Instead, explain *future* stock returns using risk factors computed from other time series.

y_{it} : trailing stock returns

x_{jt} : predictors, known at time t

f_{kt} : factors

$$y_{i,t+1} = \sum_k \beta_{ik} f_{kt} + \varepsilon_{i,t+1}$$

$$f_{kt} = \sum_j \gamma_{kj} x_{jt} + \eta_{kt}$$

Industrial forecasting with exponentially
smoothed recurrent neural networks
M. Dixon (2020)

An α -RNN is an RNN

$$h_t = \tanh(Wx_t + Uh_{t-1} + b)$$

$$y_t = Wh_t + b$$

with a smoothed hidden state

$$\tilde{h}_t = \tanh(Wx_t + Uh_{t-1} + b)$$

$$h_t = (1 - \alpha)h_{t-1} + \alpha\tilde{h}_t$$

$$y = Wh_t + b$$

(apply α coordinate-wise, and re-estimate it from time to time).

stratematch: prognostic score stratification
using a pilot design
R.C. Aikens et al.

Before (propensity score or Mahalanobis) matching, stratify the data (fit a model $y \sim x|T = 0$ on some of the controls – do not reuse these data).

Solving mixed integer programs
using neural networks
V. Nair et al.

Represent a mixed integer program (MIP) as a bipartite graph with variables and constraints as nodes, edges whenever a variable appears in a constraint, and node or edge features from the coefficients of the problem (and the solution of the relaxation) and learn two models (heuristics customized to a dataset):

- A branching model, using imitation learning to reproduce a known good (but expensive) branch-and-bound heuristic (“full strong branching”);
- A “diving” model, to generate good partial assignments.

Exact combinatorial optimization with graph convolutional neural networks

M. Gasse et al. (2019)

Imitation learning to reproduce known good heuristics; list of additional features, mostly from the solution of the relaxed problem.

A unified algorithm for the non-convex penalized estimation: the ncpen package

D. Kim et al. (2020)

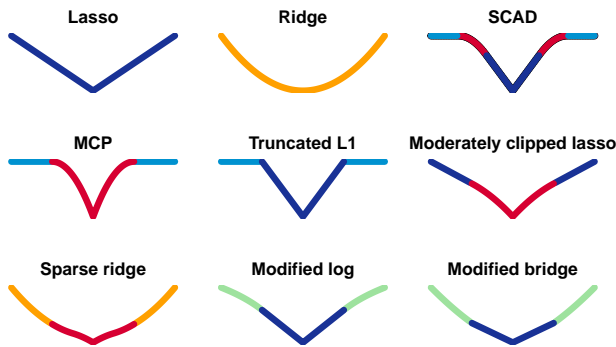
Optimization of a convex loss with a non-convex penalty can be performed with the CCCP (convex concave procedure, or difference of convex (DC) algorithm): write the objective as a difference of convex functions

$$\begin{aligned} \text{Loss}(\beta) &= L(\beta) + \sum J(|\beta_i|) \\ &= \underbrace{L(\beta) + \kappa \sum |\beta_i|}_{\text{convex}} + \underbrace{\sum D(\beta_i)}_{\text{concave}} \\ \text{Loss}(\beta) &\leq L(\beta) + \kappa \sum |\beta_i| + \sum \nabla D(\bar{\beta}_i) \beta_i \\ &= U(\bar{\beta}, \beta) \end{aligned}$$

and iteratively minimize the upper bound obtained by linearizing the concave term

$$\bar{\beta} \leftarrow \underset{\beta}{\text{Argmin}} U(\bar{\beta}, \beta).$$

To minimize U , use a quadratic approximation of $L(\beta)$ around $\bar{\beta}$, combined with line search.



ROCnReg: an R package for receiver operating characteristic curve inference with and without covariates

M.X. Rodríguez-Álvarez and V. Inácio (2020)

The ROC curve plots the true and false positive fractions

$$\begin{aligned} x &= P[Y \geq c \mid D = 0] \\ y &= P[Y \geq c \mid D = 1]. \end{aligned}$$

The *partial area under the curve* (pAUC) is the area under the curve such that $x \in [0, u_1]$ (largest acceptable FPR) or $y \in [v_1, 1]$ (lowest acceptable TPR). The *Youden index* is

$$\text{YI} = \underset{p}{\text{Max}} |\text{ROC}(p) - p|.$$

One can also consider the *covariate-specific ROC* $\text{ROC}(p|x)$ and the *covariate-adjusted* one

$$\text{AROC}(p) = \int \text{ROC}(p|x) dP(x)$$

(it is different from the *pooled ROC*).

One can compute bootstrap estimates of the ROC curve (it can be considered as a cdf) and confidence intervals for the AUC; there are also frequentist and Bayesian approaches.

Precision-recall-gain curves: PR analysis done right
P.A. Flach and M. Kull

In case of class imbalance, if there are too many negative samples, ignore the true negatives and consider the precision-recall curve.

Rescale $1/\text{precision}$ and $1/\text{recall}$, linearly, so that they span $[0, 1]$, before making the precision-recall plot and computing its area under the curve (ignore the negative gains).

$$\begin{aligned} \text{precG} &= \frac{\text{prec} - \pi}{(1 - \pi)\text{prec}} = 1 - \frac{\pi}{1 - \pi} \frac{\text{FP}}{\text{TP}} \\ \text{recG} &= \frac{\text{rec} - \pi}{(1 - \pi)\text{rec}} = 1 - \frac{\pi}{1 - \pi} \frac{\text{FN}}{\text{TP}} \end{aligned}$$

NlinTS: an R package for causality detection in time series
Y. Hmamouche (2020)

Nonlinear generalizations of Granger causality include

- Nonlinear models, *i.e.*, comparing

$$\begin{aligned} Y_t &= f(Y_{<t}) + \varepsilon_t \\ \text{and } Y_t &= g(Y_{<t}, X_{<t}) + \varepsilon_t \end{aligned}$$

where f and g are neural networks;

- Transfer entropy

$$\begin{aligned} \text{TE}_{X \rightarrow Y} &= I(Y_t; X_{<t} \mid Y_{<t}) \\ &= H(Y_t \mid Y_{<t}) - H(Y_t \mid Y_{<t}, X_{<t}) \end{aligned}$$

which can be normalized by dividing by $H(Y_t \mid Y_{<t})$.

Estimators of transfer entropy often rely on discretization. Instead, one can use nearest neighbours.

$$\begin{aligned} H(X) &= - \underset{x \sim p}{\text{E}} [\log p(x)] \\ \hat{H}(X) &= - \frac{1}{n} \sum \log \hat{p}(x_i) \\ \hat{H}(X) &= \Gamma(n) - \Gamma(k) + \log c + \frac{m}{n} \sum d_i \end{aligned}$$

(assuming that \hat{p} is locally constant in balls around each x_i , where Γ is the gamma function, m the dimension, d_i twice the distance between x_i and its k th nearest neighbour, and c the volume of the unit ball. (The estimator can be improved by letting k vary with i .)

***tsmp: an R package for time series
with matrix profile***
F. Bischoff and P.P. Rodrigues (2020)

***Model-based clustering
of multivariate ordinal data
relying on a stochastic binary search algorithm***
C. Biernacki and J. Jacques (2015)

To sample from the BOS (binary ordinal search) model for ordinal data on $\llbracket 1, m \rrbracket$ with location μ and precision π , search μ in $\llbracket 1, m \rrbracket$ with stochastic binary search:

- Start with the interval of candidates $\llbracket 1, m \rrbracket$;
- Pick an element at random in the interval of candidates;
- Compare it with μ , with probability π ; if there is no comparison, select the result with probabilities proportional to the lengths of the intervals $e_{<}$, $e_{=}$, $e_{>}$;
- Reduce the interval accordingly;
- Continue until the interval is a singleton.

It can be estimated with the EM algorithm if $m \leq 8$ (the latent variables are the pivots chosen and whether the comparisons were performed). One can also consider mixtures and/or multivariate extensions.

***OneStep:
Le Cam's one-step estimation procedure***
A. Brouste et al. (2020)

Instead of computing the MLE:

- Start with a crude estimate, e.g., from moment matching, quantile matching, or the MLE on a small sample of the data;
- Do only one Newton step

$$\theta \leftarrow \theta - I_{\theta}^{-1} \frac{1}{n} \sum \dot{\ell}(\theta, x_i)$$

using the information matrix I_{θ} if known, or an estimator

$$\hat{I}_{\theta} = -\frac{1}{n} \sum \ddot{\ell}(\theta, x_i).$$

It is faster than the MLE, and still asymptotically efficient.

***Regularized transformation models:
the tramnet package***
L. Kook and T. Hothorn

Statistical models often estimate a conditional mean: they look for a function f such that

$$f(x) = \mathbb{E}[Y|X=x].$$

Instead, *transformation models* estimate the whole conditional distribution

$$Y|X=x \sim f(\cdot|X=x).$$

They can still be linear: $f(Y) - X'\beta \sim N(0, 1)$. The transformation can be parametrized with *Bernstein polynomials*

$$b_{n,k} = \binom{n}{k} x^k (1-x)^{n-k},$$

for which monotonicity is easy to enforce ($\sum_k \theta_k b_{k,n}$, $\theta_k \leq \theta_{k+1}$; cf the **basefun** package). The models can be stratified (a different transformation for each value of a categorical variable, but the same linear parameters β).

***Skew-t expected information matrix evaluation
and use for standard error calculations***
R.D. Martin et al. (2020)

The standard skew normal distribution has density $f(x) = 2\phi(x)\Phi(\alpha x)$. The standard T distribution is the distribution of $T = Z/(\chi_{\nu}^2/\nu)$, where $Z \sim N(0, 1)$ and χ_{ν}^2 are independent. The standard skew- T distribution is the distribution of $T = Z/(\chi_{\nu}^2/\nu)$, where $Z \sim \text{SN}(0, 1, \alpha)$.

***SimilaR:
R code clone and plagiarism detection***
M. Bartoszuk and M. Gagolewski (2020)

Detect plagiarism from the *program dependence graph*: a tree, indicating how statements are nested (like the AST, but with no ordering), with “data dependence edges” constraining the order in which the statements can be executed.

In addition, canonicalize statements (e.g., spurious braces), recognize duplicated variables, remove dead code, assume functions have no side effect (and can be memoized), rewrite map-like expressions (**apply***) as loops.

***ROBustness in network (robin):
an R package for comparison
and validation of communities***
V. Policastro et al. (2020)

To assess the significance of communities in a graph, check how much perturbation is needed to alter them (titration).

***AQuadtree: an R package for quadtree
anonymization of point data***
R. Lagonigro et al. (2020)

To preserve privacy, bin spatial data in a quadtree, putting isolated points in an “isolated” bin instead of discarding them.

RNGforGPD: an R package for generation of univariate generalized Poisson data
H. Li et al. (2020)

The generalized Poisson distribution (GPD)

$$P(y) = \begin{cases} \theta(\theta + \lambda y)^{y-1} e^{-\theta - \lambda y} \\ 0 \text{ if } \lambda < 0 \text{ and } y > \lfloor -\theta/\lambda \rfloor \end{cases}$$

allows for over- and under-dispersion in count data.

Its multivariate generalization is

$$\begin{aligned} X_i &\sim \text{GPD}(\theta_i, \lambda_i) & i \in \llbracket 0, m \rrbracket \\ Y_i &= X_i + X_0 & i \in \llbracket 1, m \rrbracket \end{aligned}$$

Package wsbacfit for smooth backfitting estimation of generalized structured models
J. Roca-Pardiñas et al. (2020)

Generalized structured models (GSM)

$$\Lambda(Y) = G(Z, \beta, g(X)) + S(T, \delta, s(U)),$$

with G, S, Λ known, β, g, δ, s unknown, generalize GAMs

$$Y = \sum g_i(X_i) + \varepsilon$$

and varying coefficient models

$$Y = \sum g_i(X_i) Z_i + \varepsilon.$$

gk: an R package for the g-and-k and generalized g-and-h distributions
D. Prangle (2020)

The g-and-k and g-and-h distributions are 4-parameter univariate distributions of the form

$$\begin{aligned} X &= A + BG(Z)H(Z) \\ Z &\sim N(0, 1) \end{aligned}$$

where A and B are location and scale parameters, G introduces asymmetry, and H fattens the tails.

$$\begin{aligned} G(z) &= 1 + c \tanh(gz/2) \\ H(z) &= z(1 + z^2)^k \\ H(z) &= z \exp(hz^2/2). \end{aligned}$$

(They are only defined from their quantiles: the density is not available in closed form.)

A new versatile discrete distribution
R. Turner (2020)

The *Beta-binomial distribution*

$$\begin{aligned} p &\sim \text{Beta}(\alpha, \beta) \\ X &\sim \text{Bin}(n, p) \end{aligned}$$

only allows for over-dispersion and its maximum likelihood estimation is unreliable. Instead, try the *discretized beta*

$$p(x) \propto \text{dbeta}_{\alpha, \beta} \left(\frac{x+1}{N+2} \right), \quad x \in \llbracket 0, N \rrbracket.$$

Linear fractional stable motion with the rlfsm R package
S. Mazur and D. Otryakhin (2020)

LFSM generalizes fractional Brownian motion (fBM):

$$X_t = \int [(t+s)_+^{H-1/\alpha} - (-s)_+^{H-1/\alpha}] dL_s,$$

where L is a symmetric α -stable Lévy motion.

IndexNumber: an R package for measuring the evolution of magnitudes
A. Saavedra-Nieves and P. Saavedra-Nieves (2020)

There are many ways of combining quantities and prices over time, in a CPI-like fashion: Paasche, Marshall-Edgeworth, Laspeyres, Fisher indices.

The Lambert way to Gaussianize heavy-tailed data with the inverse of Tukey's h transformation as a special case
G.M. Goerg (2014)

Tukey's h distribution

$$\begin{aligned} U &\sim N(0, 1) \\ Z &= U \exp(\tfrac{1}{2} h U^2), \end{aligned}$$

which can be generalized to $Y = \sigma Z + \mu$, has fat tails. Conversely, one can use the inverse transformation (expressible with the Lamber W function, *i.e.*, the inverse of $u \mapsto ue^u$) to remove fat tails.

Finding optimal normalizing transformations via bestNormalize
R.A. Peterson (2020)

To make your data more Gaussian, try \log , $\sqrt{\cdot}$, \exp , Argsinh , Box-Cox (power transformation), Yeo-Johnson (Box-Cox, separately for positive and negative inputs), Lambert $W \times F$, and ordered quantiles.

R Journal (2020)

The **crqa** package (recurrence quantification analysis, RQA) extracts features from a recurrence (or cross-recurrence) plot.

For fast topological data analysis (TDA), prefer Ghudi in TDA in dimension 2, and Ripser in TDAstats in higher dimensions.

The **biglasso** package provides out-of-code lasso, with memory-mapped files and *feature screening rules*.

The **DChaos** package provides several estimators of the Lyapunov exponent.

The **ProjectManagement** package designs Gantt charts, identifies critical activities, computes early and last times of each activity, manages resources and costs – e.g., to allocate delay costs using Shapley values.

The futures API

```
f <- future(expr)
v <- value(f)
r <- resolved(f)
```

enables asynchronous and parallel processing. Also check `future.apply`, `furrr`, `doFuture`; `parallel`, `foreach`, `doMC` (Unix), `doParallel` (Windows); `parallelly`, `globals`, `listenv`.

The KSPM package provides kernel regression.

The `gofCopula` package provide all the goodness-of-fit tests you can think of, for the most common copulas.

The `ari` package generates videos from a presentation and a script, using `ffmpeg` (`tuneR`) and (paid) text-to-speech APIs (AWS Polly, Microsoft or Google).

The R-Docker ecosystem has become very large.

Seeded CCA is CCA after dimension reduction (when $p > n$, it is faster than penalized CCA).

When reshaping data (normalizing a data-frame), one may use regular expressions, e.g., `(.*)[.](.*)`; the capture groups can be named (`?<part>.*`)[.](`?<dim>.*`); named argument can simplify this syntax: `part='.*'`, `'[.]'`, `dim='.*'`. Alternatives include `reshape2`, `data.table`, `tidyr`, `tidyfast`, `tidyfst`, etc.

The TULIP package provides (six) sparse generalizations of LDA; it uses `tenstr` for tensor computations.

The NTS package provides non-linear time series models, such as threshold AR (TAR), functional AR

$$X_t = \sum_i \phi_i * X_{t-i} + \varepsilon_t,$$

where X_t , ε_t , ϕ_i are functions, and non-linear non-Gaussian state space models (sequential Monte Carlo, SMC).

The `cran2copr` package uses Redhat's `copr` buildsystem to turn CRAN packages requiring compilation into (rpm-installable) binary packages. Also check `bspm` for cross-distribution binary packages.

***Deep matrix tri-factorization:
mining vertex-wise interactions
in multi-space attributed graphs***
Y. He (2020)

Given the incidence matrix of a graph, $G \in \{0, 1\}^{n \times n}$, and node features $S \in \mathbf{R}^{n \times s}$, $O \in \mathbf{R}^{n \times o}$, matrix tri-factorization approximates the graph by finding \hat{S} , \hat{O} , W minimizing

$$\|G - \hat{S}W\hat{O}'\|^2 + \lambda_1 \|S - \hat{S}\| + \lambda_2 \|O - \hat{O}\|$$

and can help predict missing edges. The matrix W “harmonizes” the two feature spaces, linearly: instead, replace the bilinear mapping $(S, O) \mapsto SWO'$ with a neural net; this generalizes to more than two feature spaces.

***Fast nonnegative matrix tri-factorization
for large-scale data co-clustering***
H. Wang et al.

The fast nonnegative matrix factorization

$$\begin{array}{ll} \text{Find} & F \in \mathbf{R}^{d \times m} \\ & S \in \mathbf{R}^{m \times c} \\ & G \in \mathbf{R}^{n \times c} \\ \text{To minimize} & \|X - FSG'\|^2 \\ \text{Such that} & F, S, G \geq 0 \\ & F, G \in \Psi \end{array}$$

where Ψ is the set of cluster indicator matrices (*i.e.*, boolean object \times cluster matrices: $g_{ij} = 1$ if x_i belongs to cluster j). It generalizes k -means but clusters both rows and columns.

Regularize by adding terms $\text{tr}(G'L_dG)$ and $\text{tr}(F'L_fF)$ where $L = I - D^{-1/2}WD^{1/2}$ is the normalized graph Laplacian.

Scalable non-negative matrix tri-factorization
A. Čopar et al. (2017)

Matrix trifactorization

$$\text{Minimize}_{U, S, V} \|X - USV'\|^2,$$

estimated by gradient descent

$$\begin{array}{l} U \leftarrow U \odot \frac{XVS'}{USV'VS'} \\ V \leftarrow V \odot \frac{X'US}{VS'U'US} \\ S \leftarrow S \odot \frac{U'XV}{U'USV'V} \end{array}$$

possibly with orthogonality constraints $U'U = I$, $V'V = I$,

$$\begin{array}{l} U \leftarrow U \odot \sqrt{\frac{XVS'}{U'U'XVS'}} \\ V \leftarrow V \odot \sqrt{\frac{X'US}{V'V'S'US}} \\ S \leftarrow S \odot \sqrt{\frac{U'XV}{U'USV'V}} \end{array}$$

can be computed blockwise.

***Constant function market makers:
multi-asset trades via convex optimization***
G. Angeris et al. (2021)

Decentralized (crypto-currency) exchanges (DES) do not use a limit order book (LOB) but a constant function market maker (CFMM), an automated market maker implemented as a smart contract, and to which anyone can contribute liquidity (as with an ETF). Its reserves are $R \in \mathbf{R}_+^n$ (for n assets) and it maintains a vector of weights $v \in \mathbf{R}_+^n$, $\mathbf{1}'v = 1$ for the stakes of the liquidity providers. A trade (Δ, Λ) (“I give $\Delta \in \mathbf{R}_+^n$ and want $\Lambda \in R\mathbf{R}_+^n$) is accepted if $\phi(R + \Delta - \Lambda) = \phi(R)$ (or $R + \gamma\Delta - \Lambda$, $\gamma < 1$, to account for a trading fee),

for some concave, increasing, differentiable function ϕ , such as

$$\phi(R) = p'R \quad (\text{pegged: constant prices})$$

$$\phi(R) = \prod R_i^{w_i}$$

$$\phi(R) = (1 - \alpha)\mathbf{1}'R + \alpha \prod R_i^{w_i}$$

$$\phi(R) = \mathbf{1}'R - \alpha \prod R_i^{-1}.$$

The unscaled prices are

$$p_i = \frac{\partial \phi(R)}{\partial R_i};$$

the prices are p_i/p_n if asset n is the numéraire.

Inverse options on a Black-Scholes world

C. Alexander and A. Imeraj (2021)

Options on Deribit, an unregulated bitcoin exchange, are *inverse option*: options to buy or sell bitcoin, priced, margined and settled in bitcoin, which avoids the use of USD.

$$\text{Payoff}_{\text{Call}}(S) = K \cdot \text{Max}(K^{-1} - S^{-1}, 0)$$

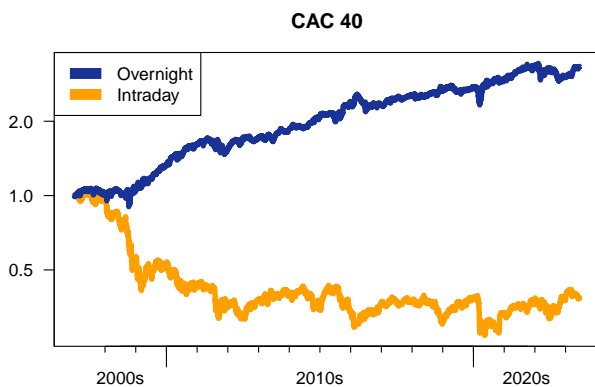
S : price of BTC in USD

K : strike in USD

They chose not to tell you

B. Knuteson (2021)

Overnight returns are less volatile, and higher, than intraday returns.



A sparsity algorithm

with applications to corporate credit rating

D. Wang et al. (2021)

When looking for a counterfactual explanation of the output of a black-box model, *i.e.*, the minimum change to the input to change the output, add a sparsifying penalty to the optimization problem, to make the explanation more actionable.

Financial return distributions:

past, present and covid-19

M. Wątorrek et al. (2021)

To model distribution tails:

$$f(x) \sim |x|^{-(1+\alpha)} \quad 0 < \alpha < 2 \quad \alpha\text{-stable}$$

$$f(x) \sim |x|^{-(1+\alpha)} e^{-\gamma|x|} \quad \gamma > 0 \quad \text{truncated } \alpha\text{-stable}$$

$$f(x) \sim |\exp|(x^{-\beta}) \quad 0 < \beta < 1 \quad \text{stretched exponential}$$

$$f(x) \sim e_q(-\beta x^2) \quad q\text{-Gaussian}$$

where the q -Gaussian distribution maximizes the Tsallis entropy if the first and second moments are fixed, and the q -exponential is

$$e_q(x) = [1 + (1 - q)x]_+^{1/(1-q)}.$$

End-to-end risk budgeting portfolio optimization with neural networks

A.S. Uysal et al. (2021)

Do not use risk budgeting blindly: use it end-to-end, to maximize the information ratio (with `cvxpylayers`):

- Start with features (returns and volatility for 7 ETFs);
- Use a neural network to compute the risk budgeting parameters, b , c ;
- Solve the risk budgeting problem

$$\begin{aligned} \text{Find} \quad & y \geq 0 \\ \text{To minimize} \quad & y' \Sigma y \\ \text{Such that} \quad & \sum b_i \log y_i \geq c \end{aligned}$$

- Compute the weights, $w = y / \|y\|_1$;
- Compute the information ratio.

The unreasonable effectiveness of optimal transport in economics

A. Galichon (2021)

Applications of optimal transport in economics include many 2-player situations, such as *matching problems*

x : worker

y : job

α_{xy} : job satisfaction

γ_{xy} : output

w_{xy} : wage

$$u_x = \text{Max}_y (\alpha_{xy} + w_{xy})_+$$

$$v_y = \text{Max}_x (\gamma_{xy} + w_{xy})_+$$

Maximize u_x and v_y

or hedonic models

x : producer
 y : consumer
 z : quality
 P_z : price
 C_{xz} : production cost
 U_{yz} : utility
 Maximize $P_z - C_{xz}$ and $U_{yz} - P_z$.

If P (resp. Q) is the martingale measure for options on X (resp. Y), the price of an option with payoff $u(X)$, resp. $v(Y)$, is $E_P u(X)$, resp. $E_Q v(Y)$. If the market is incomplete and there is no unique martingale measure for (X, Y) , the price of an option with payoff $\Phi(X, Y)$ is at most

$$\max_{\pi \in \mathcal{M}(P, Q)} E_\pi \Phi(X, Y).$$

If $P = \text{Unif}(0, 1)$, the solution of the optimal transport problem is the graph of the cdf of Q – it defines quantiles. Replacing $[0, 1]$ with $[0, 1]^n$ gives *multivariate quantiles*.

Inverse optimal transport tries to recover the transport cost Φ_{xy} from the optimal transport plan π_{xy} , often from covariates ϕ_{xyk}

$$\pi_{xy} = \exp \left[\sum_k \lambda_k \phi_{xyk} - u_x - v_y \right]$$

(λ , u , v unknown), for instance, to link import-exports π to several measures of country similarity ϕ_{xyk} .

Stock price prediction using BERT and GAN **P. Sonkiya et al. (2021)**

Use a GAN to forecast returns (e.g., from technical indicators): you get a distribution of plausible returns (as with a Bayesian model) – or an ensemble of forecasts.

Deep risk models: a deep learning solution for mining latent risk factors to improve covariance matrix estimation **H. Lin et al.**

Feed stock returns to:

- A GRU, then a FC layer;
- A GAT (graph attention network), then a GRU and a FC layer

to generate risk factor returns, which are then used linearly, to reproduce stock returns (the loss is the explained variance, and, to stabilize the model, the explained forward variance), with a penalty for multicollinearity (VIF).

Correlation scenarios and correlation stress testing **N. Packham and F. Woebbecking (2021)**

Model the correlation matrix as

$$c_{ij} = \tanh \left[\eta + \sum_k \lambda_k \mathbf{1}_{\text{only one of } i \text{ and } j \text{ is exposed to } k} + \sum_k \nu_k \mathbf{1}_{\text{both } i \text{ and } j \text{ are exposed to } k} \right]$$

where i and j are the assets, and k the risk factors.

Malliavin calculus in finance **E. Alós and D.G. Lorite (2021)**

1. The implied volatility surface is the relation

$$\sigma \sim \log(F/K) + T$$

where F is the forward price. For T fixed, it exhibits a smile, a skew; as $T \rightarrow \infty$, it becomes more symmetric and flatter; empirically, the slope is proportional to $1/\sqrt{T}$.

In incomplete markets, delta hedging can be replaced by gamma hedging (hedging Δ and Γ using bond, underlying and vanilla options) or vega hedging (hedging Δ and vega).

The implied volatility can be expressed as a weighted average of future volatilities.

$$I_0 = \frac{\int_0^T e^{rt} \sigma_t^2 \frac{\partial^2 \text{BS}}{\partial x^2} S^2 dt}{\int_0^T e^{rt} \frac{\partial^2 \text{BS}}{\partial x^2} S^2 dt}$$

2. The realized variance is an estimator of the integrated variance,

$$\int_s^t \sigma_u^2 du \approx \sum r_{t_i, t_{i+1}} \quad (\text{log-returns}),$$

but it is contaminated by microstructure noise.

Prefer the Fourier estimator of volatility

$$\sum_j (\Delta_j X)^2 + \sum_{j \neq \ell} \left(\sum_{-N}^N e^{ik(t_j - t_\ell)} \right) \Delta_j X \Delta_\ell X.$$

Spot volatility exhibits jumps, clustering, roughness and long memory (there is evidence for and against both short and long memory – one could have both, e.g., with the sum of a long-memory process and a short-memory one).

Under reasonable assumptions, a *local volatility model*

$$\frac{dS}{S} = rdt + \sigma(t, S)dW$$

consistent with observed prices exists (*i.e.*, it has the correct marginals, but unrealistic dynamics), from Gyöngi's lemma and Dupire's formula

$$\sigma^2(T, K) = \frac{\partial_T C + rK \partial_K C}{\frac{1}{2} K^2 \partial_K^2 C}.$$

For instance, the CEV (constant elasticity of variance) model

$$dS = \mu S dy + \sigma S^\gamma dW,$$

for $\gamma < 1$, ensures $\text{Cor}(S, \sigma) < 0$ and produces a downward sloping skew.

Stochastic volatility models have two sources of randomness,

$$\begin{aligned} dS &= rSdt + \sigma SdW \\ \sigma &= f(Y) \\ dY &= a(t, Y)dy + b(t, Y)dB; \end{aligned}$$

the function f can be exponential, square root, absolute value; the process Y can be log-normal, Ornstein-Uhlenbeck or CIR.

For instance, the Heston model uses the square root of a CIR process,

$$\begin{aligned} \sigma &= \sqrt{v} \\ dv &= k(\theta - v)dt + \nu\sqrt{v}dB, \end{aligned}$$

where ν is the vol-of-vol.

The SABR (stochastic alpha beta rho) model is

$$\begin{aligned} dF &= \sigma F^\beta dW & \beta \in [0, 1] \text{ (often, } \beta = 1) \\ d\sigma &= \alpha \sigma dB & \alpha > 0 \\ \rho &= \frac{d\langle W, B \rangle_t}{dt}. \end{aligned}$$

Stochastic-local volatility models combine both ideas

$$\frac{dS}{S} = rdt + \sigma_t \lambda(t, S) dW,$$

where σ is a diffusion.

Rough volatility models are stochastic volatility models driven by a fractional Brownian motion, such as the rough Bergomi model.

$$\begin{aligned} dS &= rSdt + \sigma SdW \\ \log \frac{\sigma_t^2}{\sigma_0^2} &= \nu\sqrt{2H}Z_t - \frac{1}{2}\nu^2 t^{2H} \\ Z_t &= \int_0^t (t-s)^{H-\frac{1}{2}} dB, \quad H < \frac{1}{2}. \end{aligned}$$

The variance swap payoff is

$$\sum \log \frac{S_i}{S_{i-1}} - K,$$

where K (fair price) is chosen so that the initial contract price be zero. The fair price of its continuous analogue is

$$\mathbb{E} \int_0^T \sigma_t^2 dt = 2 \mathbb{E} \log \frac{S_T}{S_0}.$$

The volatility swap payoff is

$$\sqrt{\sum \log \frac{S_i}{S_{i-1}}} - K;$$

the fair price of its continuous analogue

$$\mathbb{E} \sqrt{\int_0^T \sigma_t^2 dt}$$

is more difficult to compute.

The gamma swap is a weighted variance swap

$$\int_0^T a(u) \sigma_u^2 du$$

with $a(u) = S_u/S_0$; it can be replicated with a European option with payoff

$$\frac{S_T}{S_0} \log \frac{S_T}{S_0}.$$

For arithmetic variance swaps, $a(u) = S_u^2$.

3. Malliavin calculus is a calculus of variations for stochastic processes.

The **Malliavin derivative** is an inverse D of the operator

$$h \mapsto W(h) = \int_0^T h(s) dW_s$$

satisfying $DW(h) = h$ and $Df(A) = f'(A)DA$.

If $F = f(W(h_1), \dots, W(h_n))$, it is defined as

$$DF = \sum \partial_i f(W(h_1), \dots, W(h_n)) h_i;$$

it is then extended to the closure of the set of such functions.

For instance, for Brownian motion, $D_s W_t = \mathbf{1}_{[0,t]}(s)$.

For exponential Brownian motion, $D_r S_t = \sigma S_t \mathbf{1}_{[0,t]}(r)$.

It satisfies the following properties.

$$\begin{aligned} \mathbb{E}\langle DF, h \rangle &= \mathbb{E}[FW(h)] \\ D\psi(F_1, \dots, F_n) &= \sum \partial_i \psi DF_i \\ D(FG) &= (DF)G + F(DG) \\ \mathbb{E}[G\langle DF, h \rangle] + \mathbb{E}[F\langle DG, h \rangle] &= \mathbb{E}[FGW(h)] \\ D_r A_t &= 0 \text{ for } r > t \text{ if } A \text{ is adapted} \\ D_s \mathbb{E}_t[A] &= \mathbb{E}[D_s A] \mathbf{1}_{[0,t]}(s) \end{aligned}$$

The **divergence operator**, or **Skorohod integral**, is the adjoint of the Malliavin derivative.

$$\mathbb{E}\langle DF, u \rangle = \mathbb{E}[F\delta(u)]$$

It generalizes the Ito integral to non-adapted processes.

$$\delta(u) = \int_0^T u(s) dW(s)$$

For processes of the form $u(s) = \sum A_i h_i(s)$.

$$\delta(u) = \sum_i A_i W(h_i) - \langle DA_i, h_i \rangle.$$

In particular

$$\begin{aligned} \delta \left[\sum A_i \mathbf{1}_{[t_i, t_{i+1}]}(s) \right] &= \\ \sum A_i (W_{t_{i+1}} - W_{t_i}) - \sum \int_{t_i}^{t_{i+1}} D_s A_i ds \end{aligned}$$

where the second term is zero if the A_i are adapted (Ito integral).

For a diffusion

$$dX = \mu ds + \sigma dW$$

the Malliavin derivative is

$$DX_t = \int_0^t D\mu_s ds + \sigma \mathbf{1}_{[0,t]} + \int_0^t D\sigma_s dW_s;$$

it can be computed as the solution of a SDE or, after a change of variable, an ODE. It is easy to compute for most volatility models.

4. If F is \mathcal{F}_T -measurable, it can be expressed as (*martingale representation theorem*)

$$F = E[F] + \int_0^T m(T, s) dW_s.$$

The **Clark-Ocone-Haussman formula** gives an explicit martingale representation,

$$F = E[F] + \int_0^T E_r[D_r F] dW_r.$$

It can be used to build hedging strategies (apply it to the final value of the replicating portfolio, $V_T = \alpha_T S_T + \beta_T e^{rT}$) or price VIX options (apply it to σ_t^2).

The **integration by parts formula**

$$E[f'(F)G] = E\left[f(F)\delta\left(\frac{Gu}{\langle DF, u \rangle}\right)\right]$$

can be used to compute the expectation of a derivative (Greeks), with Monte Carlo simulations.

The **anticipating Ito formula** generalizes Ito's formula to non-adapted processes.

5. **Fractional Brownian motion** (fBM) is a Gaussian process with covariance function

$$E[B_t B_s] = \frac{1}{2}(t^{2H} + s^{2H} - |t - s|^{2H});$$

in particular, $B_t - B_s \sim N(0, |t - s|^{2H})$. For $H \neq \frac{1}{2}$, it is not Markovian, it is not a martingale, nor a semi-martingale, and the increments are correlated: $E[(B_t - B_s)(B_u - B_r)] = \dots \neq 0$. It is self-similar, $(B_{at})_t \stackrel{d}{=} (a^H B_t)_t$, and λ -Hölder continuous for $\lambda < H$.

Since it is not a semi-martingale, we cannot use Ito calculus to define stochastic integration.

It has a representation as

$$B_t = \int_0^t K(t, s) dW_s$$

$$K(t, s) \propto (t - s)^{H - \frac{1}{2}} +$$

$$\left(\frac{1}{2} - H\right) \int_s^t (u - s)^{H - \frac{3}{2}} \left(1 - \left(\frac{s}{u}\right)^{\frac{1}{2} - H}\right) du$$

The **Riemann-Liouville fractional Brownian motion** (RLfBM) truncates this kernel and only keeps the first term,

$$B_t \propto \int_0^t (t - s)^{H - \frac{1}{2}} dW_s.$$

The Malliavin derivative is $D_s B_t = K(s, t)$; one can also compute the Malliavin derivative of most rough volatility models.

Python for algorithmic trading

Y. Hilpisch (2020)

For readers not very familiar with Python.

**Artificial intelligence in finance:
a Python-based guide**

Y. Hilpisch (2020)

For readers who have not been exposed to deep learning yet (with a chapter on reinforcement learning).

**Introducing localgauss, an R package
for estimating and visualizing
local Gaussian correlation**

G.D. Berensten et al. (2014)

To assess local changes in correlation, look at:

– The correlation curve, $x \mapsto \rho(x)$,

$$\rho = \beta \frac{\sigma_X}{\sigma_Y} = \frac{\beta \sigma_X}{\sqrt{(\beta \sigma_X)^2 + \sigma_\varepsilon^2}}$$

$$\beta(x) = \frac{d}{dx} E[Y|X = x]$$

$$\sigma_\varepsilon^2(x) = \text{Var}[Y|X = x];$$

– The conditional correlation (exceedance correlation)
– but, for a Gaussian distribution it tends to 0 in the tails (unless $|\rho| = 0$);
– The Holland-Wang dependence

$$\gamma(x, y) = \frac{\partial^2}{\partial x \partial y} \log f_{X,Y}(x, y),$$

but it is not in $[-1, +1]$ – in the Gaussian case,

$$\gamma = \frac{\rho}{1 - \rho^2} \frac{1}{\sigma_X \sigma_Y};$$

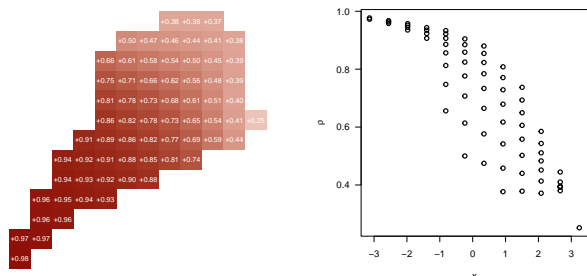
– The *local Gaussian correlation*, obtained by maximizing the *local likelihood*

f : density

ϕ_θ : Gaussian density with parameter θ

$$\text{Minimize}_{\theta_x} \int k_h(v - x) [\phi_{\theta_x}(v) - \log \phi_{\theta_x}(v) f(v)] dv$$

$$\text{i.e. } \int k_h(v - x) \nabla_\theta \log \phi_\theta(v) [f(v) - \phi_\theta(v)] dv = 0.$$



Recognizing and visualizing copulas: an approach using local Gaussian approximation
G.D. Berentsen et al. (2014)

Use Gaussian pseudo-observations to compute the local Gaussian correlations.

Dynamic copula methods in finance
U. Cherubini et al. (2012)

0. Given the joint distribution of (X, Y) , we want to describe the joint distribution (copula and margins) of $(X, Y, X + Y)$. Applications include credit risk (sum of non-Gaussian, dependent risks) and market dynamics (non-independent increments – with the added requirement that the resulting process be Markov and a martingale).

1. Correlation trading (resp. correlation risk) exploits (resp. measures the impact of) changes in correlation. Copula-based models should not introduce any spurious arbitrage, e.g., when pricing derivatives with different expiry dates, or when aggregating risk (VaR).

2. The price of a bivariate digital put, MDP (receive \$1 if $S_A \leq K_A$ and $S_B \leq K_B$) is related to the price of the univariate digital puts, DP_A, DP_B , via the copula C :

$$\begin{aligned} \text{MDP} &= vP[S_A \leq K_A, S_B \leq K_B] \\ &= vC(Q_A(K_A), Q_B(K_B)) \\ &= vC\left(\frac{DP_A}{v}, \frac{DP_B}{v}\right) \end{aligned}$$

where v is the discount factor. Other examples include: bivariate digital corridor, digital best-of, first-to-default, nth-to-default.

The *Fréchet* copulas are mixtures of the mix, max and independence copulas.

The *Kendall function* of an Archimedean copula

$$k(t) = P[C(U, V) \leq t] = t - \frac{\phi(t)}{\phi'(t)}$$

is useful to derive closed-form formulas for Archimedean copulas.

The following bivariate copula is not exchangeable:

$$(u, v) \mapsto u^\alpha v^\beta C(u^{1-\alpha}, v^{1-\beta}), \quad \alpha, \beta \in [0, 1].$$

Archimedean copulas can be made hierarchical, by replacing $C(\dots) = \phi^{-1}[\phi(\cdot) + \phi(\cdot) + \phi(\cdot) + \phi(\cdot)]$ with $\phi_3^{-1}[\phi_3 \phi_1^{-1}(\phi_1(\cdot) + \phi_1(\cdot)) + \phi_3 \phi_2^{-1}(\phi_2(\cdot) + \phi_2(\cdot))]$.

The conditional probabilities are

$$\begin{aligned} P[X \leq x | Y = y] &= \frac{\partial C(F_X(s), v)}{\partial v} \Big|_{v=F_Y(y)} \\ P[X \leq x | Y \leq y] &= \int_0^{F_Y(y)} \partial_2 C(F_X(x), w) dw. \end{aligned}$$

In particular, for a *1-factor copula*, i.e., if the X_i 's are independent given X_1 ,

$$\begin{aligned} P[U_2 \leq u_2, \dots, U_n \leq u_n | U_1 = u_1] &= \prod_{i>1} \partial_1 C(u_1, u_i) \\ C(u_1, \dots, u_n) &= \int_0^{u_1} \prod_{i>1} \partial_1 C(u, u_i) du. \end{aligned}$$

This can be generalized to more factors.

3. Copulas can be used to describe, not only cross-sectional data, but also time dependence – but we need to impose a few constraints on the result:

- The Markov property, which results from the efficient market hypothesis – the price should contain all the information;
- The martingale property: the time series should be unpredictable.

Given a Markov process $(U_t)_t$ with uniform margins, we have, for $s \leq r \leq t$,

$$\begin{aligned} P(U_s \leq u_s, U_t \leq u_t) &= \int P[U_s \leq u_s, U_t \leq u_t | U_r \leq u] du \\ &= \int P[U_s \leq u_s | U_r = u] \cdot P[U_t \leq u_t | U_r \leq u] du \\ &= \int \partial_2 C_{sr}(u_s, u) \cdot \partial_1 C_{rt}(u, u_t) du \end{aligned}$$

We can define a product a bivariate copulas:

$$A * B(u, v) = \int_0^1 \partial_2 A(u, t) \partial_1 B(t, v) dt.$$

It is associative, non-commutative, and satisfies

$$\begin{aligned} C * \Pi &= \Pi * C = \Pi && \text{(zero)} \\ C * M &= M * C = C && \text{(one)} \\ W * W &= M && \text{(minus one)} \\ W * C * W &= \hat{C} && \text{survival copula} \end{aligned}$$

More generally, $A * B(x, y) = A \star B(x, 1, y)$, where

$$A \star B(u_1, \dots, u_{m+n-1}) = \int_0^1 \partial_m A(u_1, \dots, u_{m-1}, \xi) \partial_1 B(\xi, u_{m+1}, \dots, u_{m+n-1}) d\xi.$$

A stochastic process is Markov if its copulas satisfy

$$C_{t_1, \dots, t_n} = C_{t_1, t_2} \star C_{t_2, t_3} \star \dots \star C_{t_{n-1}, t_n}.$$

It is mixing if $\lim_{k \rightarrow \infty} C_{i, i+1} = \Pi$.

It is right-continuous if $\lim_{k \rightarrow 0^+} C_{i, i+k} = M$.

Brownian dynamics are defined by $C_{st} = C_{\rho^{t-s}}^{\text{Gauss}}$.

Farlie-Gumbel-Morgenstern dynamics can only produce very low correlations.

$$\begin{aligned} C(u, v) &= uv + \theta uv(1-u)(1-v) \quad \theta \in [-1, 1] \\ \theta_{st} &= \frac{1}{3} \theta_{sr} \theta_{rt} \\ \theta_n &= 3 \left(\frac{\theta}{3} \right)^n \end{aligned}$$

Time-changed Brownian copulas are more flexible: every semi-martingale is a time-changed Brownian motion (with the quadratic variation for martingales and, more generally, Lévy subordinators or other increasing processes, such as cumulated volatility or cumulated jump intensity).

The C -convolution of F_X and F_Y is the copula of $(X, X + Y)$:

$$\begin{aligned} F_X \overset{C}{*} F_Y(u, v) &= C_{X, X+Y}(u, v) \\ &= \int_0^u \partial_1 C[w, F_Y(F_{X+Y}^{-1}(v) + F_X^{-1}(w))] dw \\ F_{X+Y}(t) &= \int_0^1 \partial_1 C[w, F_Y(t - F_X^{-1}(w))] dw. \end{aligned}$$

A Markov process $(X_t)_t$ is a martingale iff

- (i) The increments $X_t - X_s$ have finite mean;
- (ii) $\forall s, t \forall u \in [0, 1]$ almost everywhere

$$\int_0^1 F_{X_t - X_s}^{-1}(v) \partial_1 C_{X_s, X_t - X_s}(u, dv) = 0.$$

This is the case, for instance, if the increments and the copula are symmetric: $\tilde{C}(u, v) := u - C(u, 1 - v) = C(u, v)$.

4. The c -quantile curve of y conditioned on x , $q(x, p)$, is the value of y such that

$$p = F(y|x) = \partial_1 C(F_X(x), F_Y(y)).$$

A *stationary copula-based Markov process* is defined by its invariant distribution G^* and the copula C of (C_{t-1}, X_t) . To estimate it, use the empirical distribution for G^* , and the MLE for the copula C .

The α -mixing coefficients are

$$\alpha_k = \sup_{m \in \mathbf{Z}} \sup_{\substack{A \in \mathcal{F}_{-\infty}^m \\ B \in \mathcal{F}_{m+k}^\infty}} |P(A \cap B) - P(A)P(B)|.$$

The β -mixing coefficients are similar, but use partitions $\{A_1, \dots, A_I\}$, $\{B_1, \dots, B_J\}$ of Ω and $\sum_{ij} |P(A_i \cap B_j) - P(A_i)P(B_j)|$.

The ρ -mixing coefficients are

$$\rho_k = \sup_{m \in \mathbf{Z}} \sup_{\substack{f \in L^2(\mathcal{F}_{-\infty}^m) \\ g \in L^2(\mathcal{F}_{m+k}^\infty)}} \text{Cor}(f, g).$$

The maximum correlation of a copula C is

$$\rho_C = \sup_{\substack{f, g \in L^2(0,1) \\ \int f = \int g = 0 \\ \int f^2 = \int g^2 = 1}} \iint f(u)g(v)C(du, dv).$$

if $\rho_C < 1$, then a stationary Markov process defined from C is α - and ρ -mixing, i.e., $\alpha_k \rightarrow 0$, $\rho_k \rightarrow 0$ when $k \rightarrow \infty$. This is the case for the Student and Marshall-Olkin copulas.

A process $(X_t)_t$ has long memory if $H(t, h) \sim_{h \rightarrow \infty} Ah^{-p}$, $A > 0$, $p > 0$, and short memory if $H(t, h) = O(e^{-Ah})$, $A > 0$, where the Hellinger measure of dependence between X_t and X_{t+h} is

$$H(t, h) = 1 - \iint f_{X_t X_{t+h}}^{1/2}(x, y) f_{X_t}^{1/2}(x) f_{X_{t+h}}^{1/2}(y) dx dy.$$

It can be computed from the copulas $\frac{1}{2} \iint (c^{1/2} - 1)^2 - 1$ and replaced by other measures of dependence: $\iint c^2 - 1$, $\iint c \log c$, $\iint C(u, v) - uv$, $\iint (C(u, v) - uv)^2$, $\sup |C(u, v) - uv|$.

We can also define non-parametric Markov processes (i.e., processes with functional parameters), for instance,

$$X_t = \begin{cases} X_{t-1} + \varepsilon_t & \text{with probability } \eta(X_{t-1}) \\ \varepsilon_t & \text{otherwise.} \end{cases}$$

Analyzing dependent data with vine copulas: a practical guide with R C. Czado (2019)

1. Distribution F , density f , copula C , copula density c , conditional density and h -functions are related as follows.

$$\begin{aligned} F(x_1, \dots, x_d) &= C(F_1(x_1), \dots, F_d(x_d)) \\ f(x_1, \dots, x_d) &= f(F_1(x_1), \dots, F_d(x_d)) f_1(x_1) \cdots f_d(x_d) \\ c(u_1, \dots, u_d) &= \frac{\partial^d C(u_1, \dots, u_d)}{\partial u_1 \cdots \partial u_d} \\ f(x_1|x_2) &= c_{12}(F_1(x_1), F_2(x_2)) f(x_2) \\ F(x_1|x_2) &= \frac{\partial C(F_1(x_1), u_2)}{\partial u_2} \Big|_{u_1=F_2(x_2)} \\ h_{1|2}(u_1|u_2) &= \frac{\partial C(u_1, u_2)}{\partial u_2} \end{aligned}$$

2. Some dependence measures can be computed from the copula.

$$\begin{aligned} \rho &= 12 \iint_{[0,1]^2} uv dC(u, v) - 3 \\ \tau &= 4 \iint_{[0,1]^2} C(u, v) dC(u, v) - 1 \\ \lambda_L &= \lim_{t \rightarrow 0^+} \frac{C(t, t)}{t} \end{aligned}$$

3. Bivariate copula classes include: elliptical (Gaussian, Student), Archimedian (defined from a generator function, $C(u, v) = \phi^{-1}(\phi(u) + \phi(v))$, Clayton, Gumbel, Frank, Joe, a few 2-parameter families (Clayton-Gumbel (BB1), Joe-Gumbel (BB6), Joe-Clayton (BB7), Joe-Frank (BB8)), Tawn (3 parameters, with shapes not symmetric along the diagonal), Marshall-Olkin, etc.

Extreme value copulas are the limiting copulas of $(\text{Max}_{1 \leq i \leq n} X_i, \text{Max}_{1 \leq i \leq n} Y_i)$; they are the copulas satisfying

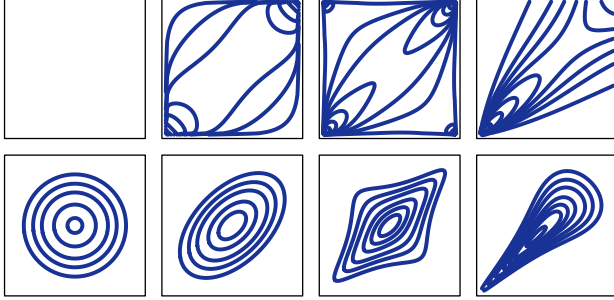
$$\forall m \in \mathbf{N}^\times \quad \forall u, v \in [0, 1] \quad C(u, v) = C(u^{1/m}, v^{1/m})^m.$$

They are the copulas of the form

$$C(u, v) = \exp \left[\log(uv) A \left(\frac{\log v}{\log(uv)} \right) \right]$$

where $A : [0, 1] \rightarrow [\frac{1}{2}, 1]$ is called the *Pickands dependence function*.

Copulas can be plotted in x -scale (original), u -scale (uniform margins) or z -scale (Gaussian margins).



To sample from a bivariate copula:

$$\begin{cases} u \sim \text{Unif} \\ v \sim h_{2|1}(\cdot|u) \end{cases} \quad i.e. \quad \begin{cases} u \sim \text{Unif} \\ \varepsilon \sim \text{Unif} \\ v = h_{2|1}^{-1}(\varepsilon|u). \end{cases}$$

To estimate copula parameters, first process the margins (parametrically or not – this may involve regression or time series modeling) and invert the formula giving Kendall's τ from the ccopula parameter.

Let $C_{12|3}$ be the conditional distribution of U_1, U_2 given U_3 . The margins are no longer uniform: it is not a copula. The margins are $C_{1|3}$ and $C_{2|3}$: the *conditional bivariate copula* is

$$C_{12;3}(u_1, u_2|v_3) = C_{12|3}(C_{1|3}^{-1}(u_1|v_3), C_{2|3}^{-1}(u_2|v_3)).$$

Conditional copulas depend on the conditioning value, but it can be averaged out:

$$C_{12;3}^A(u_1, u_2) = \int C_{12;3}(u_1, u_2|v_3) dv_3.$$

It is a copula, which can also be obtained as the *partial copula*, i.e., the distribution of (these are uniform)

$$\begin{aligned} V_{1|3} &= C_{2|3}(U_2|U_3) \\ V_{2|3} &= C_{1|3}(U_1|U_3). \end{aligned}$$

4. A 3-variate copula can be decomposed using bivariate copula (pair copula construction, PCC) by repeatedly using $f_{1|2} = c_{12}f_1$:

$$\begin{aligned} f_{123} &= f_{1|23}f_{2|3}f_3 \\ f_{2|3} &= c_{23}f_2 \\ f_{1|23} &= c_{12;3}f_{1|3} \\ &= c_{12;3}c_{13}f_1 \\ f_{123} &= c_{12;3}c_{13}c_{23}f_1f_2f_3. \end{aligned}$$

The conditional copula $c_{12;3}$ depends on x_3 , but we will assume it does not (*simplified PCC*).

$$\begin{aligned} f_{123}(x_1, x_2, x_3) &= c_{12;3}(F_{1|3}(x_1|x_3), F_{2|3}(x_2|x_3); x_3) \cdot \\ &\quad c_{13}(F_1(x_1), F_3(x_3)) \cdot \\ &\quad c_{23}(F_2(x_2), F_3(x_3)) \cdot \\ &\quad f_1(x_1)f_2(x_2)f_3(x_3) \end{aligned}$$

Conversely, we can define a 3-variate copula density as

$$\begin{aligned} c(u_1, u_2, u_3) &= c_{12;3}(C_{1|3}(u_1|u_3), C_{2|3}(u_2|u_3)) \cdot \\ &\quad c_{13}(u_1, u_3)c_{23}(u_2, u_3) \end{aligned}$$

where $C_{1|2}(u_1|u_2) = \partial C_{12}/\partial u_2$. The copula c_{12} can be computed by integration.

In dimension d , there are many possible decompositions.

$$\begin{aligned} c_{1\dots d} &= \prod_{i < j} c_{ij:i+1, \dots, j-1} & \text{D-vine} & \quad \text{Diagram: A chain of nodes connected sequentially.} \\ c_{1\dots d} &= \prod_{i < j} c_{ij:1, \dots, i-1} & \text{C-vine} & \quad \text{Diagram: A central node connected to multiple other nodes.} \end{aligned}$$

5. A **regular vine** (R-vine) is a sequence of trees T_1, \dots, T_{d-1} such that

- (i) $V(T_1) = \llbracket 1, d \rrbracket$;
- (ii) $V(T_i) = E(T_{i-1})$;
- (iii) $\{a, b\} \in E(T_i) \implies |a \cap b| = 1$.

To simplify the notations, let $A_e = \text{union}(\text{flatten}(e))$ be the complete union and $D_{\{a,b\}} = A_a \cap A_b$ the conditioning set.

A *D-vine* is a vine whose trees are chains.

A *C-vine* is a vine whose trees are stars.

Regular vines can model multivariate Gaussian or Student distributions, with the partial correlations as parameters.

Regular vines can be encoded in triangular matrices.

6. To sample from a multivariate distribution:

$$\begin{aligned} w_1 &\stackrel{\text{iid}}{\sim} U(0, 1) \\ x_1 &= F_1^{-1}(w_1) \\ x_2 &= F_2^{-1}(w_2|x_1) \\ &\vdots \\ x_d &= F_d^{-1}(w_d|x_1, \dots, x_{d-1}) \end{aligned}$$

7. To find a good starting point for the optimization, start to estimate the parameters sequentially. For instance, for $c_{123} = c_{12|3}c_{13}c_{12}$, first estimate c_{12} , c_{13} , then estimate $c_{12|3}$ from the *pseudo-observations* $C_{1|3}(u_{k1}|u_{k3}), C_{2|3}(u_{k2}|u_{k3})$ (for observation k).

8. If the vine structure is known, but not the copula families, select them with the AIC, sequentially or not.

If the vine structure is not known, use a greedy approach (Dißmann algorithm), with a minimum spanning tree for each tree, computed from $|\tau|$, AIC, a copula gof p -value, etc. For C-vines, it suffices to choose

the root of the tree. For D-vines, it suffices to choose the order of the nodes (a Hamiltonian path, with a TSP solver).

9. To compare models, use the AIC, BIC, KLIC (KL criterion – log-likelihood) or the likelihood ratio test (with correction for model complexity).

10. With stock return data: fit univariate GARCH(1,1) models with Student innovations and use the residuals; find a representative stock in each sector (high $\sum_i |\tau_{ij}|$); fit a C-vine copula to each sector (`VineCopula::RVineStructSelect`); use it for portfolio VaR computation (you need parametric margins).

11. Recent developments in vine copula based modeling include:

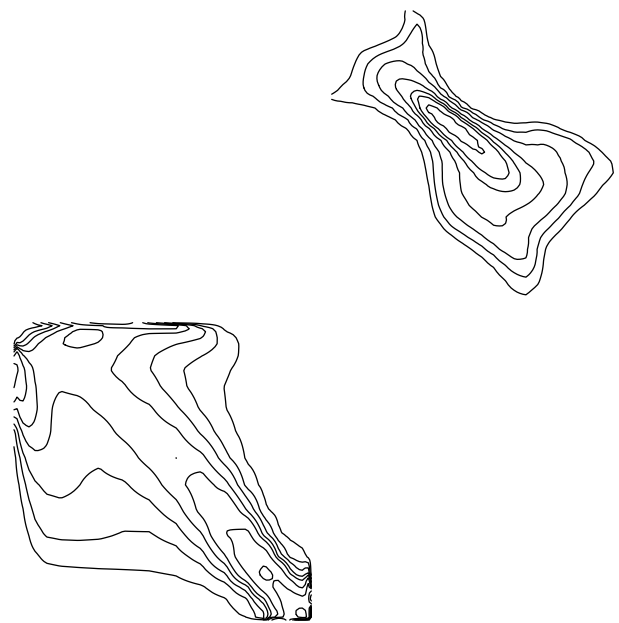
- Bayesian vine copulas: they provide credible intervals;
- Intependence tests, for the pair copulas, to simplify the models (you can also truncate them);
- Factor vine copulas (?);
- Non-parametric pair copulas;
- Non-simplified vines (with covariates);
- Tests for constant conditional correlation, to select the vine structure violating the simplifying assumption as little as possible;
- Goodness-of-fit (gof) tests;
- Approximate KL divergence between vine copulas;
- Relation between graphical models and vine copulas (conditional independences correspond to independence copulas);
- Conversely, vine copulas inside graphical models;
- Discrete variables;
- D-vine-copula-based quantile regression;
- Factor structure for vine copulas.

Applications include:

- Copulas with a time-varying dependence structure;
- D-vines for longitudinal data;
- Modeling the relations between the entries of realized variance matrices (parametrized with the logarithm of the Cholesky decomposition);
- Medical diagnostic test accuracy (copula of TP, TN, P);
- Survival data;
- Copulas with zero inflation (in insurance).

For software implementations, check `VineCopula` (supercedes `CDVine`) and `vinecopulib` (C++/R/Python).

```
## Error in file(file, "rt"): cannot open
the connection
## Error in `[.xts`(d, , 1:10): subscript
out of bounds
```



Simulating copulas: stochastic models, sampling algorithms and applications J.F. Mai and M. Scherer (2017)

1. Bivariate copulas can be sampled as $(C_{1|2}^{-1}(U_1|U_2), U_2)$, where $U_1, U_2 \sim U(0, 1)$ and $C_{1|2}(u_1|u_2) = \partial C / \partial u_2$.

Measures of dependence can be computed from the copula:

$$\tau = 1 - 4 \iint \frac{\partial C}{\partial u_1} \frac{\partial C}{\partial u_2}$$

$$\rho = \frac{\iint C - \iint \Pi}{\iint M - \iint \Pi}$$

where $\Pi(\mathbf{u}) = \prod u_i$ is the independence copula and $M(\mathbf{u}) = \min_i u_i$ the comonotonicity copula.

For an *exchangeable copula*, the correlation (the same for all pairs) satisfies $\rho \geq -1/(d-1)$.

An **extendible** sequence of random variables is part of an infinite exchangeable sequence: the correlation is then nonnegative, but it need not have positive orthant dependency,

$$P[\forall i \ X_i \geq x_i] \geq \prod_i P[X_i \geq x_i].$$

Homogeneous mixtures, $\theta \sim F$, $X_i \stackrel{\text{iid}}{\sim} F_\theta$, are exactly the extendible sequences (DeFinetti).

Heterogeneous mixtures partition the dimensions and use a different distribution for each part.

Extreme value distributions $C(u_1^t, \dots, u_d^t) =$

$C(u_1, \dots, u_d)^t$ have a Pickand representation

δ : measure on the simplex Δ_d

$$P(w_1, \dots, w_d) = \int_{\Delta_d} \text{Max}\{w_1 u_1, \dots, w_d u_d\} \delta(du)$$

$$C(u_1, \dots, u_d) = \left(\prod u_i \right)^{P(\log \mathbf{u} / \mathbf{1}' \log \mathbf{u})}.$$

2. With the sampling procedure

$$M \sim F$$

$$X_1, \dots, X_d \sim \text{Exp}(M),$$

the cdf of the margins is $\phi(x) = E[e^{-xM}]$, i.e., the Laplace transform of F , and the survival copula of (X_1, \dots, X_d) is $C(u_1, \dots, u_d) = \phi(\phi^{-1}u_1 + \dots + \phi^{-1}u_d)$ (archimedean copula). If ϕ is not completely monotonic, the same formula may still define a copula for small values of d , but it may not be extendible, just exchangeable.

Archimedean copulas can be rescaled:

- An α -stable Lévy subordinator Λ_m stopped at M , defines a new random variable $\tilde{M} = \Lambda_M$ with Laplace transform $\phi(x^\alpha)$ (the Laplace transform of a Lévy subordinator is $\psi(x) = x^\alpha$);
- The Laplace transform of $M_1 + \dots + M_\beta$ is $\phi(x)^\beta$.

Hierarchical (**H-extendible**) archimedean copulas are defined as

$$C_{\phi_0}(C_{\phi_1}(u_{11}, \dots, u_{1d_1}), \dots, C_{\phi_J}(u_{J1}, \dots, u_{Jd_J})),$$

provided the generators are compatible: ϕ_j and $(\phi_0^{-1} \circ \phi_j)'$ completely monotonic. They can be sampled as

$$E_{ji} \sim \text{Exp}(1)$$

M with Laplace transform ϕ

$\Lambda^{(1)}, \dots, \Lambda^{(J)}$ Lévy subordinators with exponents ψ_j

$$U_{ji} = (\phi_0 \circ \psi_j)(E_{ji} / \Lambda_M^{(j)}).$$

Archimedean copulas can be made asymmetric, e.g.,

$$V \sim C_\phi$$

$$\tilde{U} \sim \Pi$$

$$U_k = \text{Max}\{V_k^{1/\alpha_k}, \tilde{U}_k^{1/\alpha_k}\}$$

3. The lack of memory characterizing exponential distributions, $P[X \geq x + y | X \geq y] = P[X \geq x]$, generalizes, in dimension d , to the *Marshall-Olkin distribution*

$$\begin{aligned} \bar{F}(x_1, \dots, x_n) &= P[\forall i \ X_i \geq x_i] \\ &= \exp \left[- \sum_{\substack{I \subset \llbracket 1, d \rrbracket \\ I \neq \emptyset}} \lambda_I \cdot \text{Max}_i x_i \right]. \end{aligned}$$

There are $2^d - 1$ parameters. It can be interpreted as the time-to-failure, when the failure of a group of items $I \subset \llbracket 1, d \rrbracket$ has a different cause for each I .

To sample from it:

$$E_I \sim \text{Exp}(\lambda_I) \text{ for each } I \subset \llbracket 1, d \rrbracket, \ I \neq \emptyset$$

$$X_k = \text{Min}\{E_I : I \ni k\}.$$

The corresponding survival copula is

$$\hat{C}(u_1, \dots, u_d) = \prod_I \text{Min}_{k \in I} u_k^{\lambda_I / \sum_{j \ni k} \lambda_j}.$$

They can be sampled as

$$Y_i \sim \text{Cat}(\mathcal{P}(\llbracket 1, d \rrbracket) \setminus \{\emptyset\})$$

$$P(Y_i = I) = \lambda_I / \sum \lambda_J$$

$$\varepsilon_i \sim \text{Exp}(\sum \lambda_J)$$

$$X_k = \varepsilon_1 + \varepsilon_2 + \dots + \varepsilon_{\text{Min}\{i : k \in Y_i\}}.$$

The MO copula is an extreme value copula (but the MO distribution is not an extreme value distribution), has a singular component, has an interpretation in terms of shocks (insurance, credit risk), and positive upper tail dependence (but lower tail independence).

For an exchangeable Marshall-Olkin copula, $|I| = |J| \Rightarrow \lambda_I = \lambda_J$ and

$$C(u_1, \dots, u_d) = \prod u_{(k)}^{a_k - 1}$$

where $\forall i, j \ (-1)^i \Delta^i a_j \geq 0$ (d-monotone sequence).

To sample from it:

- Sample the time until the first shock (the minimum of exponential variables is still exponential);
- Sample the number of components affected;
- Iterate.

If the sequence $(a_k)_k$ can be extended to an infinite completely monotone sequence (with $a_0 = 1$, this is the case iff $a_k = E[X]$, for some random variable X with values in $[0, 1]$; equivalently, the Hankel determinants are nonnegative), this is an extendible MO copula.

A completely monotone sequence defines a Lévy subordinator with $\psi(1) = 1$ and a *Lévy-frailty copula* $C_\psi(u_1, \dots, u_d) = \prod u_{(i)}^{\psi(i) - \psi(i-1)}$.

To sample from it:

$$E_k \sim \text{Exp}(1)$$

$$\Lambda_t \sim \text{Lévy}(\psi)$$

$$X_k = \inf\{t > 0 : \Lambda_t > E_k\}$$

$$U_k = \exp(-X_k).$$

There is an h-extendible variant.

4. Spherical distributions can be recognized from their characteristic function $\phi_X(t) = E[e^{it'X}] = \varphi(\|t\|^2)$ (but not all univariate functions define a characteristic function: there are fewer as the dimension increases). Extendible spherical distributions are mixtures of Gaussians, $\sqrt{W}Z$, $Z \sim N(0, I)$.

For elliptical distributions $X \sim E_d(\mu, \Sigma, \phi)$, $\phi(t) = e^{it'\mu}\phi(t'\Sigma t)$; they are extendible if $\mu = \mu_1 \mathbf{1}$ and

$$\Sigma = \sigma^2 \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

h-extendible elliptic copulas have a block correlation matrix.

7. Variance can be reduced with antithetic variables ($\mathbf{u} \mapsto 1 - \mathbf{u}$), control variates (e.g., from an independence structure – it may not work well), or importance sampling (e.g., by increasing the dependence).

8. Exchangeable Marshall-Olkin copulas

$$C(u_1, \dots, u_d) = \prod u_{(k)}^{a_k - 1}$$

can be generalized to *Shock copulas*

$$C(u_1, \dots, u_d) = \prod g_k(u_{(k)}).$$

A random vector (X_1, \dots, X_d) has a *min-stable exponential distribution*, i.e., $\text{Min}\{c_1 X_1, \dots, c_d X_d\}$ have univariate exponential distributions for all $c_1, \dots, c_d \in [0, \infty]$, iff the margins are exponential and the copula is an extreme value copula.

Extendible extreme value copulas are characterized by

$$C(u_1, \dots, u_d) = \mathbb{E} \left[\exp - \sum_{k=1}^d \Lambda_{-\log(u_k)} \right]$$

for a strong IDT subordinator Λ with $\psi(1) = 1$.

A. A Lévy subordinator is a càdlàg stochastic process $(\Lambda_t)_{t \geq 0}$ with $\Lambda_0 = 0$ satisfying:

(i) Stochastic continuity (no deterministic jumps)

$$\forall t \geq 0 \forall \varepsilon > 0 \lim_{t \rightarrow 0} P[|\Lambda_{t+h} - \Lambda_t| \geq \varepsilon] = 0$$

(ii) Independent increments;

(iii) Stationary increments: $\Lambda_{t+h} - \Lambda_t \stackrel{d}{=} \Lambda_h$;

(iv) $t \mapsto \Lambda_t$ is a.s. non-decreasing.

Λ_t is infinitely divisible.

The Laplace transform of a Lévy subordinator is $\mathbb{E}[e^{-x\Lambda_t}] = e^{-t\psi(x)}$ (Lévy-Khinchin) where $\psi(x) = \mu x + \int_{[0, \infty]} (1 + e^{-tx}) \nu(dt)$ (ψ is the Laplace exponent, μ the drift, ν the Lévy measure).

Examples include:

- Poisson: $\psi(x) = \lambda(1 - e^{-x})$, $\nu(B) = \lambda \mathbf{1}_{1 \in B}$ (number of jumps of size in B per unit of time);
- Compound Poisson: $\Lambda_t = \mu t + \sum_{i=1}^{N_t} J_i$;
- Gamma: $\mu = 0$, $\nu(dt) = \beta(e^{-\eta t}/t) \mathbf{1}_{t > 0} dt$, $\psi(x) = \beta \log(1 + x/\eta)$, $\Lambda_t \sim \Gamma(\beta t + \eta)$;
- Inverse Gaussian

$$\Lambda_t \sim \text{IG}(\beta t, \eta)$$

$$\Lambda_t = \inf\{s > 0 : \eta s + X_s = \beta t\}$$

X_t standard Brownian motion

$$\alpha\text{-stable: } \nu(dt) = \frac{\alpha}{\Gamma(1-\alpha)} \frac{1}{t^{1+\alpha}} \mathbf{1}_{t > 0} dt.$$

An *additive subordinator* has independent, but not necessary stationary, increments.

A *strong IDT subordinator* is a non-decreasing càdlàg stochastic process $(\Lambda_t)_t$ with $\Lambda_0 = 0$, strongly infinitely divisible such that

$$(\Lambda_t)_{t \geq 0} = (\Lambda_{t/n}^{(1)} + \dots + \Lambda_{t/n}^{(n)})_{t \geq 0}$$

where $\Lambda^{(i)}$ are iid copies of Λ .

A partial correlation vine based approach for modeling and forecasting multivariate volatility time series N. Barthel et al. (2018)

Partial correlations on the edges of a vine provide an unconstrained parametrization of a correlation matrix.

Model volarilities and partial correlations with a HAR (heterogeneous AR, e.g., 1-, 5- and 22-day lags) or VARFIMA model.

Pair copula construction for financial applications: a review K. Aast (2016)

Applications of the pair copula construction (PCC) include:

- VaR and CVaR of a portfolio;
- Probability of default;
- Liquidity risk (bid and ask spread of several assets);
- Systemic risk (CDS of financial institutions, stock market indices);
- (Scenario-based) CVaR portfolio optimization (C-vine with skewed Student margins)
- Option pricing.

kdecopula: an R package for the kernel estimation of bivariate copula densities T. Nagler (2017)

To reduce the boundary effect when estimating a non-parametric copula density:

- Reflect the data over the sides of the square;
- Use kernels supported on the square, e.g., Beta;
- Transform the margins to make them Gaussian.

Evading the curse of dimensionality in nonparametric density estimation with simplified vine copulas T. Nagler and C. Czado (2016)

(Truncated, simplified) vine copulas provide scalable high-dimensional density estimators, by reducing the problem to a sequence of bivariate density estimations.

Implementation: `rvinecopulib`, `kde1d`.

ESG, risk and (tail) dependence
R. Bax et al. (2021)

For assets j , in sector S , in ESG category $k \in \{A, B, C, D\}$, estimate a vine copula with fixed first and second-level trees

$$\begin{array}{c} \diagup \diagdown \\ A \\ | \\ \geq B - S - D \leq \\ | \\ C \\ \diagup \diagdown \\ j \end{array} \quad \begin{array}{cccc} A, S - B, S - C, S - D, S \\ \diagup \diagdown & \diagup \diagdown & \diagup \diagdown & \diagup \diagdown \\ & j, C & & \end{array}$$

and aggregate measures of dependence (Kendall's τ) or tail dependence (tail dependence coefficient λ) as

$$\frac{|\tau_{j,k}|}{\sum_{\{j,k\} \subset i} |\tau_i|}.$$

Market making via reinforcement learning
T. Spooner et al. (2018)

Define the reward as P&L penalized against speculation (inventory \times change in price). For the state, use the inventory, the spread, the price change, the book imbalance, the signed volume, the volatility, and the RSI, with *tile coding* – combine several discretizations of the state space, e.g., from grids with different offsets. Use TD (SARSA) with eligibility traces; train on market data. The actions are

$$\begin{array}{ccccccccc} _ & _ & _ & _ & _ & _ & _ & _ & _ \\ | & | & | & | & | & | & | & | & | \\ _ & _ & _ & _ & _ & _ & _ & _ & _ \end{array}$$

and an inventory-clearing order.

**Reinforcement learning
for optimized trade execution**
Y. Nevmyvaka et al. (2006)

**Risk-sensitive compact decision trees
for autonomous execution
in presence of simulated market response**
S. Vyetenko and S. Xu (2021)

Do not maximize $E[G]$ but $E[G] - \lambda \text{Var}[G]$.

**Persistent clustering
and a theorem of J. Kleinberg**
G. Carlsson and F. Mémoli (2008)

Single linkage clustering is the only functor from the category of finite metric spaces and non-decreasing maps to the category of persistent sets (multiscale clusterings (X, θ) , there $\theta(r)$ is interpreted as the partition of X at scale r) that

- Preserves the underlying set
- Gives the expected persistent set for 2-point spaces $\Delta_2(\delta)$: $\theta(r)$ is $\{\{*\}, \{*\}\}$ is $r < \delta$ and $\{\{*, *\}\}$ otherwise;
- Gives the discrete partition for $r \ll 1$.

Coarser categories allow more such multiscale clustering functors:

- Increasing monic maps: density-based clustering;
- Isometries: anything (too general).

Classifying clustering schemes
G. Carlsson and F. Mémoli (2010)

There is no clustering scheme $(X, d) \mapsto P \in \text{Partitions}(X)$ satisfying Kleinberg's axioms: scale invariance, surjectivity (all clusterings are possible, if we change the metric), consistence (reducing/increasing intra/inter-cluster distances does not change the clustering) – we need multi-scale clusterings.

Single-linkage clustering can be defined from 2-simplices $\Delta_2(\delta)$, $\delta > 0$, and morphisms $\Delta_2(\delta) \rightarrow X$. The 2-simplex $\Delta_2(\delta)$ can be replaced by other simplices ($\Delta_m(\delta)$ is very similar to DBSCAN) or an arbitrary family of metric spaces.

**Characterization, stability and convergence
of hierarchical clustering methods**
G. Carlsson and F. Mémoli (2010)

A dendrogram (X, θ) defines an ultrametric $(x, y) \mapsto \text{Min}\{r \geq 0 : x \text{ and } y \text{ belong to the same component of } \theta(r)\}$. Conversely, an ultrametric defines a dendrogram. Single linkage clustering corresponds to the maximal subdominant ultrametric.

The distortion of a map of metric spaces $f : (X, d) \rightarrow (Y, d)$ is

$$\text{dis}(f) = \text{Max}_{x, x' \in X} |d(x, x') - d(fx, fx')|.$$

The joint distortion of $f : X \rightarrow Y$ and $g : Y \rightarrow X$ measures the degree to which they are inverses,

$$\text{dis}(f, g) = \text{Max}_{\substack{x \in X \\ y \in Y}} |d(x, gy) - d(fx, y)|.$$

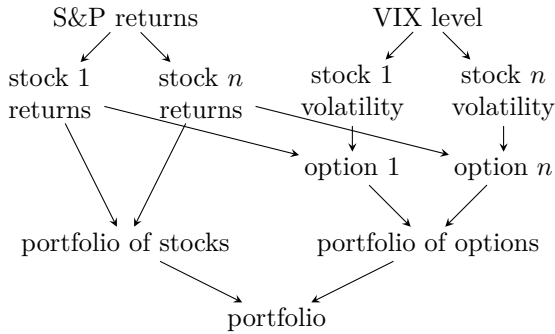
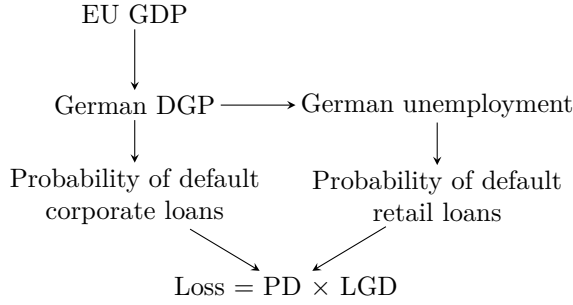
The *Gromov-Hausdorff distance* between (X, d) and (Y, d) is

$$\frac{1}{2} \text{Min}_{f, g} \text{Max}\{\text{dis } f, \text{dis } g, \text{dis}(f, g)\}.$$

Single linkage hierarchical clustering $(X, d) \rightarrow (X, u)$ is a contraction for the Gromov-Hausdorff distance. Average linkage and complete linkage are not continuous.

**A probabilistic graphical model approach
to model interconnectedness**
A. Denev et al. (2017)

The output of a model is often used as input to another: PGMs can account for those dependencies – they are not needed if you just want expectations, but the probability distributions contain more information.



Building probabilistic causal models using collective intelligence O. Laudy et al. (2021)

CausalityLink (commercial) extracts data from text:

- Numeric indicators (GDP, etc.), from a large ontology, with timestamps and duration;
- Trends: changes in those indicators over some time span;
- Events, with start and end date;
- Causal links: indicator/trend/event \rightarrow indicator/trend.

Use those links to build a Bayesian network:

- Start with the nodes you are interested in;
- Add edges between them, starting with the strongest ones, but only if you do not introduce a cycle.

Use two states (up, down); estimate the root node probabilities from the past 90 days (with a 14-day half-life); estimate the single-cause probabilities $P(B|A)$ from count data. For multiple causes, combine $P(B|A_i)$ into $P(B|A_1, \dots, A_k)$ using a log-linear model

$$P(A_1, \dots, A_k, B) \propto \exp \left[\mu + \sum \lambda_{A_\ell=i_\ell} + \lambda_{B=j} + \sum_{B=j} \lambda_{A_\ell-i_\ell} \right],$$

where $i_\ell, j \in \{\text{UP}, \text{DOWN}\}$.

Foundations of reinforcement learning with applications in finance A. Rao and T. Jelvis (2021)

Reinforcement learning (RL), or *stochastic control* is the study of sequential optimal decisions under uncertainty. The environment is described by a *Markov decision process* (MDP), *i.e.*, the conditional distribution of reward r and next state s' given current state s and action a ,

$$P(r, s'|s, a);$$

we want to find the *policy* $\pi(a|s)$ maximizing the (expected) return, *i.e.*, the discounted sum of future rewards $G_t = \sum_{s \geq t} \gamma^{s-t} R_t$.

One can distinguish between

- *Prediction*: Computing the value function $V^\pi(s) = E[G_0|S_0 = s]$, for a given policy π ;
- *Control*: computing the optimal value function V^* , and the corresponding optimal policy π^* ;
- *Planning*: solving a known MDP (often, with dynamic programming);
- *Learning*: solving an unknown MDP.

1. *Inventory management* can be modeled as an MDP:

- State: number of items on inventory, constrained to be at most N ;
- Action: number of items to order (there might be a (fixed) delivery delay);
- Stochastics: demand $\sim \text{Poisson}(\lambda)$.

Dynamic pricing for end-of-season products can also be modeled as an MDP” assume the demand is $\text{Poisson}(\lambda)$, with λ a known function of price.

3. The *Bellman policy operator* $B^\pi : V \mapsto R + \gamma PV$ is a contraction: it has a fixed point. *Policy iteration* iterates it to compute V^π . *Value iteration* iterates the *Bellman optimality operator* $B^* : V \mapsto \text{Max}_a(R + \gamma PV)$: it alternates two steps,

$$\begin{aligned} V &\leftarrow V^\pi && \text{policy evaluation} \\ \pi &\leftarrow \text{greedy}(\pi) && \text{policy improvement.} \end{aligned}$$

6. *Merton’s portfolio problem* is an analytic solution to the dynamic asset allocation and consumption problem.

7. The *optimal exercise* of an American option in the binomial tree model can be computed with dynamic programming; this generalizes to more general *optimal stopping* problems.

In a (no-arbitrage) incomplete market, option prices are not uniquely defined. *Superhedging* computes upper and lower bounds, but they are often too far apart. Instead, one can compute the price maximizing some expected utility, *e.g.*, CARA (expected utility *indifference pricing*).

8. *Optimal execution* can be modeled as an MDP:

- State: price, number of shares left to sell, time left;
- Action: number of shares sold;
- Reward: utility of the sales proceeds, after correcting the price with the temporary price impact (“eating into the book”);
- Stochastics: the price follows a random walk, corrected with the permanent price impact.

For (unrealistic) linear price impacts, we can solve the Bellman equation.

8. *Optimal market making* can be modeled as an MDP:

- State: mid-price, current P&L, inventory;
- Actions: bid and ask prices and quantities;
- Reward: utility of P&L and inventory value;
- Stochastics: random walk for the mid-price; random number of shares bought and sold.

9. Many RL prediction algorithms are based on the update rule

$$V(S_t) \leftarrow V(S_t) + \alpha(\text{target} - V(S_t))$$

for different targets:

- Monte Carlo: G_t ;
- TD (temporal difference): $R_{t+1} + \gamma V(S_{t+1})$;
- n -step TD, $G_{t,2} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$, or, more generally, $G_{t,k}$;
- TD(λ), an exponentially-weighted average of the $G_{t,k}$, which can be efficiently computed (without waiting for the end of the episode) with *eligibility traces*.

10. For control, if the MDP is unknown, we need to replace the state value function V with the state-action value function Q , and alternate policy evaluation (Monte Carlo, TD, TD(λ) – for Q , TD is called SARSA), and policy improvement (ϵ -greedy rather than greedy, to explore enough).

But this is *on-policy*. If the target and behaviour policies, π and μ are different (*off-policy*):

- Q-learning uses $R_{t+1} + \gamma \text{Max}_a Q(S_{t+1}, a)$ as target;
- Importance sampling multiplies the SARSA updates with $\pi(S_t, A_t) / \mu(S_t, A_t)$ (for one step).

Convergence is not guaranteed if we combine bootstrapping (using V itself in the target), off-policy learning, and function approximation.

Which trading agent is best? Using a threaded parallel simulation of a financial market changes the pecking order

M. Rollins and D. Cliff (2020)

Automated trading algorithms (AA, GDX, ZIP, ZIC)) are often compared in market (LOB) simulators (e.g., BSE, the Bristol Stock Exchange), but which comes ahead depends on the mix of strategies they compete with. The computation time, and the resulting delay of those algorithms, may change the rankings.

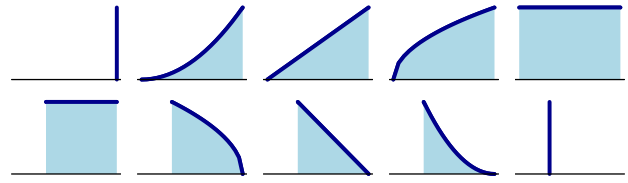
Parametrized response zero-intelligence (PRZI) traders
D. Cliff (2021)

In the *zero-intelligence constrained* (ZIC) strategy, each trader has a side (buy or sell), a limit price λ , only trades one share, and samples their bid or ask uniformly in $[p_{\min}, \lambda]$ or $[\lambda, p_{\max}]$.

The SHVR strategy bids one tick above the current best bid (if below λ).

The GVWY strategy bids at the trader's limit price (no profit, unless the trade crosses the spread).

Those strategies can be blended.



Strategic bidding in continuous double auctions
P. Vyteling et al. (2018)

The AA (adaptive aggressiveness) strategy tunes its aggressivity (eager to trade, even for lower profits, vs eager to profit, even for a lower probability of trade). Prior algorithms include:

- ZIP (zero-intelligence plus), a variant of ZIC which progressively updates its profit margin;
- GD, which estimates the probability that a bid will be accepted;
- GDX, which also uses dynamic programming, and may delay trading, waiting for better market conditions.

BSE: a minimal simulation of a limit-order-book stock exchange
D. Cliff (2018)

Reinforcement learning with expert trajectory for quantitative trading
S. Chen et al. (2021)

Reinforcement learning with expert trajectories uses (some variant of Q -learning with) a reward of 1 for the expert and 0 for the other agent's actions.

An empirical assessment of characteristics and optimal portfolios
C.G. Lamoureux and H. Zhang (2021)

Do not optimize a sum of squared residuals, but a utility function.

$$\sum_t \frac{(1 + r_{\text{port},t})^{1-\gamma}}{1-\gamma} \quad \text{utility}$$

$$r_{\text{port},t} = \sum w_{i,t-1} r_{i,t} \quad \text{portfolio returns}$$

$$w_{i,t} = f_{\theta}(x_{i,t}) \quad \text{portfolio weights}$$

$$x_{i,t} \quad \text{stock characteristics}$$

Constructing long-short stock portfolios with a new listwise learn-to-rank algorithm
X. Zhang et al. (2021)

Learning-to-rank models for information retrieval

$$\text{Loss} = \sum_{\text{relevance}} f_k \cdot \frac{1}{\underset{\substack{\text{discount} \\ \text{factor}}}{\log(1+k)}}$$

focus on best-ranked items: adapt them for stock selection by focusing on both the top and bottom items by generalizing the Plackett–Luce probability distribution on \mathcal{S}_n (ListMLE)

$$P[\sigma] = \prod_i \frac{\theta_{\sigma^{-1}(i)}}{\sum_{k \geq i} \theta_{\sigma^{-1}(k)}}$$

to consider long-short pairs (instead of long-only items)
 $\theta_i \rightsquigarrow \theta_{2n+1-i}$.

***Imputation, estimation
and missing data in finance***
G. DiCesare (2006)

Comparisons of estimators of the parameters of a partially-observed multivariate Brownian motion:

- EM algorithm;
- Impute-solve:
 - Simulate $X_{\text{mis}}|X_{\text{obs}}, \theta_k$;
 - Estimate $\hat{\theta}$ from the completed dataset;
 - Update $\theta_{k+1} = \theta_k + \gamma_k(\hat{\theta} - \theta_k)$;
- Impute-posterior:
 - Sample $X_{\text{mis}}|X_{\text{obs}}, \theta_k$;
 - Sample $\theta|X_{\text{obs}}, X_{\text{mis}}$.

The EM algorithm can be implemented with the *sweep operator*

$$\text{sweep}_k : \begin{cases} \mathbf{R}^{p \times p} & \longrightarrow \mathbf{R}^{p \times p} \\ G & \longmapsto H : h_{kk} = -1/g_{kk} \\ & h_{jk} = h_{kj} = g_{jk}/g_{kk} \\ & h_{j\ell} = h_{\ell j} = g_{j\ell} - \frac{g_{jk}g_{j\ell}}{g_{kk}} \end{cases}$$

$$\text{sweep}_1 \cdots \text{sweep}_p G = -G^{-1} \det G$$

on the sufficient statistic

$$T = \begin{pmatrix} 1 & \mathbf{1}'Y \\ Y'\mathbf{1} & Y'Y \end{pmatrix} \quad \Theta = \begin{pmatrix} -1 & \mu \\ \mu\Sigma & \end{pmatrix}.$$

Some of those computations can be generalized to SDEs: for instance, after Euler discretization, one can sample from $X_i|X_{i-1}, X_{i+1}$ with Metropolis-Hastings.

From the Markovian property, we can cut the data into blocks along dates for which all the variables are observed.

Dynamic style analysis and applications
M. Markov et al. (2004)

Return-based style analysis is a constrained regression of a fund's returns against benchmarks,

$$R_t = \alpha + \sum_i \beta_i r_{it} + \varepsilon_t, \quad \beta \geq 0, \quad \beta'\mathbf{1} = 1.$$

Allow for time-changing exposures β_{it} and add a “smoothness” penalty on $\sum \|\beta_{it} - \beta_{i,t-1}\|^2$ (smoothness would be a penalty on $\Delta^2\beta$, not $\Delta\beta$; one could also use a Kalman filter).

Choose the scale of the penalty to maximize R^2 .

To find the position of the most likely breakpoint, compare all the models with a penalty on

$$\sum_{\substack{i,t \\ t \neq t_0}} \|\beta_{i,t} - \beta_{i,t-1}\|^2,$$

for all values of t_0 (one could try the fused lasso instead).

Conditional distribution in portfolio theory
E. Qian and S. Gorman (2001)

The simplified Black-Litterman framework can be generalized for views on returns, volatilities and correlations. In the case of two assets, r_1, r_2 , and views $\tilde{\mu}_2 = E[r_2]$ and $\tilde{\sigma}_2^2 = \text{Var}[r_2]$, regress $r_1 \sim r_2$:

$$r_1 = \mu_1 + \rho \frac{\sigma_1}{\sigma_2} (r_2 - \mu_2) + \varepsilon$$

$$\varepsilon \sim N(0, \sigma_2^2(1 - \rho^2))$$

and compute $E[r_1]$, $\text{Var}[r_1]$, $\text{Cov}[r_1, r_2]$ as a function of $\rho, \mu_1, \mu_2, \sigma_1, \sigma_2, \tilde{\mu}_2, \tilde{\sigma}_2$.

For a view on $d = r_1 - r_2$, regress r_1 and r_2 against d .

In the general case $r \sim N(\mu, \Sigma)$ with views on $v = Pr$, regress $r \sim v$:

$$r = \mu + \Sigma P' \Sigma_v^{-1} (r - \mu_v) + \varepsilon$$

$$\varepsilon \sim N(0, \Sigma - \Sigma P' \Sigma_v^{-1} P \Sigma)$$

$$\mu_v = P \mu$$

$$\Sigma_v = P \Sigma P'.$$

The posterior is then

$$\tilde{\mu} = \mu + \Sigma P' \Sigma_v^{-1} (\tilde{\mu}_v - \mu_v)$$

$$\tilde{\Sigma} = \Sigma + \Sigma P' (\Sigma_v^{-1} \tilde{\Sigma}_v \Sigma_v^{-1} - \Sigma_v^{-1}) P \Sigma.$$

Global portfolio optimization revisited: a least discrimination alternative to Black-Litterman
J. P  zier (2007)

Given a prior distribution p (e.g., $X \sim N(\mu, \Sigma)$) and constraints on a posterior distribution \tilde{p} (e.g., $E[PX] = v$, as in Black-Litterman, or $PX = v$, or $\text{Var}[PX] = V$), look for the posterior distribution \tilde{p} , satisfying those constraints, and minimizing the difference in certainty equivalent of the optimal (dynamic) strategy (Black-Litterman only considers static strategies, *i.e.*, portfolios)

$$\text{CE}(f_{\tilde{p}}(x) | \tilde{p}) - \text{CE}(f_p(X) | p).$$

For a payoff (random variable) Y and a utility function u , the certainty equivalent is defined as $E[u(Y)] = u(\text{CE}[Y])$.

Fully flexible views: theory and practice

A. Meucci (2008)

Given a prior distribution on asset returns (and/or risk factors), and views, *i.e.*, constraints on some of the generalized moments, blend them:

- Take N samples from the prior; see them as a uniform distribution p (you could also use importance sampling);
- Solve the optimization problem

Find q , probability distribution on the samples

To minimize the relative entropy $\mathcal{E}(q, p) = \int q \log \frac{q}{p}$

Such that the constraints be satisfied

- If there are confidence levels on the views, use the mixture $(1 - c)p + cq$ (or $\sum c_i q_i$ if there are several views);
- Use for portfolio construction.

In the Gaussian case:

$$X \sim N(\mu, \Sigma)$$

$$E[QX] = \tilde{\mu}$$

$$\text{Var}[GX] = \tilde{\Sigma}$$

$$X \sim N(\bar{\mu}, \bar{\Sigma})$$

$$\bar{\mu} = \mu + \Sigma Q' (Q \Sigma Q')^{-1} (\tilde{\mu} - Q \mu)$$

$$\bar{\Sigma} = \Sigma G' [(Q \Sigma Q')^{-1} \tilde{\Sigma} (Q \Sigma Q')^{-1} - (Q \Sigma Q')^{-1}] G \Sigma.$$

In the general case, to impose constraints on expectations, volatilities and correlations, use

$$\begin{aligned} \sum_j q_j X_{jk} &= m_k \\ \sum_j q_j X_{jk}^2 &= \hat{m}_k^2 + \sigma_k^2 \\ \sum_j q_j X_{jk} X_{j\ell} &= \hat{m}_k \hat{m}_\ell + \hat{\sigma}_k \hat{\sigma}_\ell c_{k\ell} \end{aligned}$$

(this is not quite right, because we use \hat{m} and $\hat{\sigma}$ from the prior, but it makes the constraints linear and the problem convex).

Deep learning for portfolio optimization

Z. Zhang et al. (2020)

Neural net (LSTM), fed past prices and returns to directly compute the weights maximizing the information ratio (for 4 assets).

Enhancing time series momentum strategies using deep neural networks

B. Lim et al. (2019)

Time series momentum strategies require both a trend estimator and position sizing (*volatility scaling*), to

keep the volatility constant), e.g.,

$$\text{trend} = \text{past returns} \quad \text{trend} = \text{MACD}/\sigma$$

$$\text{position} = \text{sign}(\text{trend}) \quad \text{position} = \phi(\text{trend})$$

$$\phi(y) = ye^{-y^2} = \sqrt{\lambda}$$

Instead, use a neural net (WaveNet, fed the past 5 days of returns) to directly find the position size maximizing the information ratio.

Deep reinforcement learning for trading

Z. Zhang et al. (2020)

Comparison of 3 reinforcement learning algorithms, critic-only (DQN), actor-only (policy gradient) and actor-critic, to maximize the (after-transaction-cost) P&L of a (volatility-adjusted) strategy using 1m, 2m, 3m, 1y normalized returns, price, MACD(63,252), RSI(30): DQN works better.

Macroeconomic forecasting with statistically validated knowledge graphs

S. Tilly and G. Livan (2021)

To forecast/nowcast industrial production:

- Use the *global database of events, language and tone* (GDELT, available via Google BigQuery) to build theme co-occurrence graphs;
- Filter the themes relevant to economic growth with an LSTM, trained on 1000 manually labeled events (4000 nodes, 1 million edges);
- Compute its backbone using a *disparity filter*: normalize the weights of the edges connected to a given node so they sum up to 1 and apply the Benjamini-Hochberg (BH) procedure;
- Compute the *portrait divergence* to compare the graph with the previous month's;
- Compute the eigenvector centralities of all the nodes;
- Reduce the dimension of the centralities with PLS;
- Add the Baltic Dry index (a shipping index) and crude oil price to the predictors;
- Conclude with Granger causality, and a factor-augmented AR model.

An information-theoretic, all-scales approach to comparing networks

J.P. Bagrow and E.M. Bollt (2019)

The **network portrait** of a graph is a matrix $B_{\ell,k}$ = number of nodes having k nodes at distance ℓ . The 0th row contains the number of nodes; the first row is the degree distribution; each row defines a probability distribution.

```
## Error in count(.): Argument 'x' is not a vector: list
```

```
## Error in count(.): Argument 'x' is not a vector: list
```

```
## Error in count(.): Argument 'x' is not a vector: list
```

For weighted graphs, compute all shortest paths and bin their lengths

The *network portrait divergence* compares graphs using the Jensen-Shanon divergence between the distributions $P(k, \ell) \propto k B_{\ell, k}$.

Portraits of complex networks
J.P. Bagrow et al. (2007)

Comparing methods for comparing networks
T. Tantardini et al. (2019)

Distances between networks include:

- Distance between the similarity matrices $S = (I + \varepsilon^2 D - \varepsilon A)^{-1}$, where A is the adjacency matrix and D the degree;
- Distances between graphlet counts; one can also compute them on each egonet, aggregate them (Net-Dis), and use PCA;
- Distance between the graphlet correlation matrices, obtained by counting, for each node, in how many graphlets of each type it is, and computing the correlation matrix between those counts;
- Distance between the spectra of the Laplacians, e.g., via the heat signature trace $h(t) = \sum_i e^{-\lambda_i t}$;
- Graph kernels, TDA, portrait divergence, etc.

The graphlet correlation distance (GCD) gives the best results.

Open-source cross-sectional asset pricing
A.Y. Chen and T. Zimmerman (2021)

List of 300+ investment signals. Unusable code (Stata) and data (commercial).

The most general methodology to create a valid correlation matrix for risk management and option pricing
R. Rebonato and P. Jäckel (1999)

Correlation matrices can be parametrized as $C = BB'$, where the rows of B are on the sphere \mathbf{S}^{n-1} .

$$b_{ij} = \cos \theta_{ij} \prod_{k=1}^{j-1} \sin \theta_{ik} \quad 1 \leq j \leq n-1$$

$$b_{in} = \prod_{k=1}^{n-1} \sin \theta_{ik}$$

$$\theta \in \mathbf{R}^{n \times (n-1)}$$

The closest correlation matrix can be obtained by solving

$$\text{Minimize } \left\| \hat{C} - C \right\|_F^2.$$

As a starting point for the optimization:

- Eigenvalue decomposition $C = SAS'$;
- Zero out the negative eigenvalues: Λ_+ ;
- Compute $B_0 = S\sqrt{\Lambda_+}$;

- Normalize the rows of B_0 : B .

If C was a valid correlation, we have $C = BB'$.

Unconstrained parametrization for variance covariance matrices
J.C. Pinheiro and D.M. Bates (1996)

Generating realistic stock market order streams
J. Li et al. (2020)

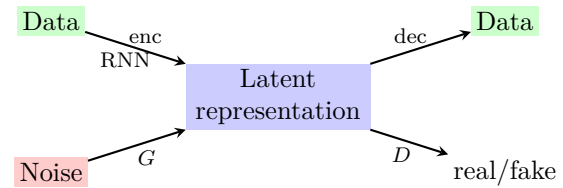
WGAN whose generator uses an LSTM to condense past orders and a (pretrained) neural net to differentially approximate the continuous double auction (CDA) mechanism.

Style transfer with time series: generating synthetic financial data
B. da Silva and S.S. Shi (2019)

Generate high frequency data with a denoising autoencoder (?) rather than a GAN or VAE; then use style transfer to get daily data.

Time series generative adversarial networks
J. Yoon et al. (2019)

Jointly learn a latent representation and a GAN.



On unbiased simulations of stochastic bridges conditional on extrema
A. Schaug and H. Chandra (2019)

The promises and pitfalls of machine learning for predicting stock returns
E. Leung et al. (2021)

Nonlinear models, e.g., GBM, have better predictive power on future returns than linear models, but their advantage almost disappears if we account for transaction costs, microcaps, multiple testing, non-synchronous implementation, risk constraints.

Matrix evolutions: synthetic correlations and explainable machine learning for constructing robust investment portfolios
J. Papenbrock et al. (2021)

Financial correlation matrices have the following properties:

- The distribution of correlations is positively skewed;
- Eigenvalues follow the Marchenko-Pastur distribution but for a very large eigenvalue (market) and a couple of large eigenvalues (industries);
- The first eigenvector has positive entries (Perron-Frobenius);

- The correlation has a “hierarchical structure”, *i.e.*, the distances from the correlation $\sqrt{1 - c_{ij}}$ are very close to the distances on the (single linkage) dendrogram d_{ij} ; this can be measured by the *cophenetic correlation* $\text{Cor}(\sqrt{1 - c}, d)$;
- The minimum-spanning tree is scale-free, *i.e.*, it has a power law degree distribution.

To generate real-looking correlation matrices, pick matrices at random in some neighbourhood of the empirical correlation and use multiobjective optimization (why?) to match

- The average correlation;
- The Gini coefficient of the eigenvalues;
- The single-linkage cophenetic correlation;
- The sum of the negative coefficients of the first eigenvector.

Those matrices can then be used to compare investment strategies, *e.g.*, risk parity and hierarchical risk parity.

MOEA/D: A multi-objective evolutionary algorithm based on decomposition
Q. Zhang and H. Li (2007)

To solve the multiobjective optimization problem

$$\text{Maximize}_{x \in \Omega} f_1(x), \dots, f_n(x),$$

reduce it to a set of 1-dimensional problems

$$\text{Maximize}_{x \in \Omega} \sum_i \lambda_i f_i(x)$$

(where λ is fixed) and solve them sequentially, starting each optimization with the solution of the closest λ .

Instead, MOEA/D solves the problems simultaneously:

- P_j : Maximize $\sum_{x \in \Omega} \lambda_{ij} f_i(x)$ subproblems;
- $x_{j,t}$: best solution to P_j after t steps;
- $x_{j,t+1}$: computed from the $x_{j',t}$ for the subproblems where $\lambda_{j'}$ is close to λ_j

Contrary to domination-based multiobjective optimization, decomposition-based algorithms can deal with higher-dimensional problems (if there are many objectives, most candidate solutions are non-dominated).

History, shocks and drifts: a new approach to portfolio formation
M. Kritzman and D. Turkington (2021)

Generate scenarios by changing low-frequency components (shocks, trend) and keeping the high-frequency part.



Spectral temporal graph neural network for multivariate time series forecasting
D. Cao et al. (2021)

To forecast multivariate time series, stack the following layers:

- GRU: $X \mapsto R$;
- *Correlation layer*, *i.e.*, self-attention

$$W = \text{Softmax}\left(\frac{QK'}{\sqrt{d}}\right), \quad Q = RW_1, \quad K = RW_2;$$

use the attention matrix W as a graph incidence (or weight) matrix in the next layer;

- Graph Fourier transform, independently for each component;
- 1-dimensional convolution;
- Gated linear unit: $\text{GLU}(a, b) = a \odot \sigma(b)$;
- (Repeat several times)
- Fully-connected layer.

Fast ES-RNN: a GPU implementation of the ES-RNN algorithm
A. Redd et al. (2019)

ES-RNN uses an LSTM to compute the parameters of a Holt-Winters exponential smoothing model; it won the M4 competition – Theta, another exponential smoothing variant, won the previous one.

COIN: Compression with implicit neural representation
E. Dupont et al. (2021)

To compress an image, overfit a neural net (pixel coordinates \mapsto RGB) to it and store the quantized weights; it is competitive with JPEG at low rates.

Learning continuous image representation with local implicit image function
Y. Chen et al. (2020)

From an image $H \times W \times 3$, compute features $H \times W \times D$. In the neighbourhood of a pixel (h, w) , use the features $M_{h,w,\cdot}$ of that pixel and those $M_{h \pm 1, w \pm 1, \cdot}$ of its neighbours to forecast RGB values. Since a point is in several neighbourhoods, ensemble their forecasts, using areas as weights.

Learning neural network subspaces
M. Wortsman et al. (2021)

Do not look for a point in weight space but a subspace (segment, Bézier curve, simplex): in each epoch, pick a random point in the subspace to train, and force the endpoints to be orthogonal; at test time, use the mid-point.

A unified approach to measurement error and missing data: overview and applications
M. Blackwell et al. (2017)

Multiple overimputation generalizes multiple imputation to noisy observations – a missing observation is an observation with infinite noise.

imputeTS:
time series missing value imputation in R
S. Moritz and T. Batrz-Beielstein (2017)

Missing values in univariate time series can be imputed by a Kalman filter (ARIMA or structural models), interpolation, or seasonal decomposition (or lof, mean, median, mode, moving average, constant value, random value).

Comparison of different methods for univariate time series imputation in R
S. Moritz et al. (2015)

The zoo and forecast packages provide several `na.*`; also check `Amelia`, `mtsdi`, `VIM`, `mice`, `imputeR` (they deal with multivariate non-time-series data: add polynomials of time, lags and leads).

Imputation of multivariate time series data: performance benchmark for multiple imputation and spectral techniques
J. Bauer et al. (2013)

First look for missingness patterns: proportion of missing data, run lengths, etc. (CDS data, 1000 tickers, 10 maturities).

Data imputation with empirical orthogonal functions (DINEOF – EOF is another term for left and right singular vectors) uses the SVD to recover the missing data, progressively increasing the number of singular values retained.

Singular spectrum analysis (SSA) adds lags

$$\begin{pmatrix} x_1 & x_2 & \cdots & \\ \vdots & \vdots & & \vdots \\ x_L & x_{L+1} & \cdots & x_N \end{pmatrix}$$

and reconstructs the signal by averaging over the anti-diagonals. It can be generalized to multivariate SSA (MSSA).

Temporal regularized matrix factorization for high-dimensional time series prediction
H.F. Yu and N. Rao (2016)

Matrix completion can be used to model time series, in high dimension, with missing values: $Y \approx FX$.

$$\text{Minimize}_{F,X} \sum_{int} (y_{it} - f'_i x_t)^2 + \lambda_1 R(F) + \lambda_2 R(X)$$

The regularizer $R(X)$ could come from a graph

$$R(X) = \sum_t \|x_t\|^2 + \sum_{s,t} w_{st} \|x_t - x_s\|^2$$

if the weights are chosen beforehand. Instead, use the negative log-likelihood of some time series model

$$R(X) = -\log P(x_1, \dots, x_T | \theta),$$

e.g., an AR model

$$R(X) = \sum_{t>L} \left\| x_t - \sum_{\ell=1}^L \theta_\ell x_{t-\ell} \right\|^2 + \sum_t \|x_t\|^2$$

and a penalty (prior) for θ .

Applications include missing value imputation and forecasting.

Probabilistic sequential matrix factorization
O.D. Akyildiz et al. (2019)

In the matrix factorization problem

$$\text{Minimize}_{C,X} \|Y - CX\|,$$

add a prior on C (matrix variate Gaussian) and X (nonlinear state space model).

$$s \sim \text{IG}$$

$$C \sim \text{MN}(C_0, I, sV_0)$$

$$x_0 \sim N(\mu_0, sP_0)$$

$$x_k \sim N(f_\theta(x_{k-1}), sQ_0)$$

$$y_k \sim N(Cx_k, sR_0)$$

(For s constant, it is Gaussian, but if we marginalize it, we get more robust Student distributions.)

Mixtures, EM and missing data
B. Stewart (2017)

We want to estimate $\mu = E[X]$ and $\Sigma = \text{Var}[X]$ in presence of missing data.

0. Start with initial estimates, e.g, means and variances (diagonal Σ) from available observations.

1. For each missingness pattern, estimate a regression $X_{\text{mis}} = X_{\text{obs}}\beta + \varepsilon$. We can compute β and $\sigma^2 = \text{Var} \varepsilon$ from μ, Σ . We this replace each missing value with a distribution $X_{\text{mis},i} \sim N(X_{\text{obs},i}\beta, \sigma^2)$. More precisely, we will need the expectations $E[X_{\text{mis},i}]$ and $E[X_{\text{mis},i}X'_{\text{mis},i}]$ (E-step).

2. Using the expectations $E[X]$ and $E[XX']$, re-estimate μ, Σ , with MLE (M-step).

3. Iterate until convergence.

Amelia II: a program for missing data
J. Honaker et al. (2011)

To impute missing data, the EMB (EM with bootstrap) algorithm:

- Bootstraps missing data $m = 5$ times
- Models each bootstrap dataset as a Gaussian $N(\mu, \Sigma)$, using the EM algorithm for MLE with missing values;
- Imputes the missing values with their conditional expectation (m times).

***What to do about missing values
in time series cross-section data***
J. Honaker and G. King (2010)

To allow smooth interpolation when needed, add basis functions to the data (e.g., with cubic splines); also add lags and leads of the variables. Implementation in *Amelia*.

mice:
multiple imputation by chained equations in R
S. van Buuren and K. Groothuis-Oudshoorn (2011)

Iterate T times:

- Sample θ_1 from $\theta_1|Y_1^{\text{obs}}, Y_2, \dots, Y_p$;
- Sample Y_1^{mis} from $Y_1^{\text{mis}}|\theta_1, Y_1^{\text{obs}}, Y_2, \dots, Y_p$;
- ...
- Sample θ_p ;
- Sample Y_p^{mis} .

Those conditional models (often, regression or GLMs) need not be compatible: they may not come from a well-defined joint distribution – this does not seem to be a problem. Generate 5 such implied datasets; fit your model on each of them; pool those models. The *VIM* package provides a few more plots.

***Gaussian process imputation
of multiple financial time series***
T. de Wolff et al. (2020)

Missing data in time series (4 to 10 time series) can be imputed with a multi-output Gaussian process (MOGP), with a multioutput spectral mixture (MOSM) kernel.

***MOGPTK:
the multioutput Gaussian process toolkit***
T. de Wolff et al. (2020)

Built with PyTorch (formerly Tensorflow/GPflow).

***Spectral mixture kernels
for multioutput Gaussian processes***
G. Parra and F. Tobar (2017)

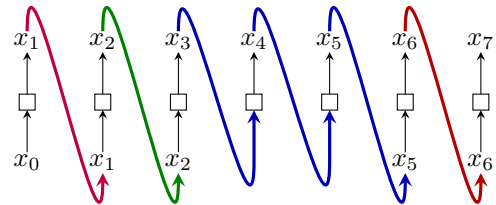
$$k_{ij}(\tau) = \sum_q \alpha_{ij}^q \exp\left[-\frac{1}{2}(\tau + \theta_{ij}^q)' \Sigma_{ij}^q (\tau + \theta_{ij}^q)\right] \cdot \cos[(\tau + \theta_{ij}^q)' \mu_{ij}^q + \phi_{ij}^q]$$

***GP-VAE:
deep probabilistic time series imputation***
V. Fortuin et al. (2019)

VAE with a low-dimensional GP prior on the latent space, with a Cauchy kernel (it can be seen as a Gamma mixture of RBF kernels at different scales). When computing the ELBO, only include terms with data. To generate data, replace the missing values with 0 (this introduces bias).

***BRITS: bidirectional recurrent imputation
for time series***
W. Cao et al. (2018)

Use an RNN (BiLSTM) to impute missing values; the loss is delayed until we have the next observation



One can add cross-sectional features.

***Recurrent neural networks for multivariate
time series with missing values***
Z. Che et al. (2017)

To deal with missing values in a RNN (GRU), use the last available value and add to the input: a missingness indicator, and the time since the last observation

***NAOMI: Non-autoregressive multiresolution
sequence imputation***
Y. Liu et al. (2019)

Deep generative models for sequences are autoregressive: errors compound. Instead, use a multi-resolution approach: feed the time series (with the missing values zeroed out) and the mask to a BiLSTM, predict $x_{n/2}$ from h_0 and h_n , feed that new value to the network, and proceed recursively.

***Multivariate time series imputation
with generative adversarial networks***
Y. Luo et al. (2018)

Train a GAN to generate complete data that looks like real data (assuming you have several/many similar time series); find a point in latent space (noise) that gives a time series close to the actual data; use it to generate a complete time series.

***E² GAN: end-to-end
generative adversarial network***
Y. Luo et al. (2019)

Use a denoising autoencoder as generator.

***Estimating missing data
in temporal data streams using
multidirectional recurrent neural networks***
J. Yoon et al. (2017)

First interpolate within streams (BiRNN), then impute across streams (AE with dropout).

**Time series imputation and prediction with
bidirectional generative adversarial networks**
M. Gupta and R. Beheshti (2020)

Given many similar time series, train a GAN whose generator fills in the missing values, and whose discriminator distinguishes between real and imputed values.

**Panning for gold: Model-X knockoffs for
high-dimensional controlled variable selection**
E. Candès et al. (2017)

To select relevant variables in a model $Y \sim X$ while controlling the false discovery rate (FDR), one could use *conditional randomization*, i.e., repeatedly sample X_i^* from $X_i|X_{-i}$, compute test statistics T_i and T_i^* , compare them, and use

$$P = \frac{1 + \sum_k \mathbf{1}_{T_i > T_i^*}}{1 + K}.$$

This can be done with a single sample ($K = 1$) with a *knockoff distribution*, i.e., a distribution (X, \tilde{X}) such that

- (i) $(X, \tilde{X}) \stackrel{d}{=} (X, \tilde{X})_{\text{swap}(S)}$, where $S \subset \llbracket 1, p \rrbracket$ and $\text{swap}(S)$ swaps X_j and \tilde{X}_j for all $j \in S$
- (ii) $\tilde{X} \perp\!\!\!\perp X | Y$ (e.g., \tilde{X} is constructed without looking at Y)

For a Gaussian distribution, $X \sim N(0, \Sigma)$, we can use

$$(X, \tilde{X}) \sim N\left(0, \begin{pmatrix} \Sigma & \Sigma - \text{diag}(s) \\ \Sigma - \text{diag}(s) & \Sigma \end{pmatrix}\right)$$

and sample from $\tilde{X}|X$, where s is chosen so that the matrix be positive semi-definite; we then have $\text{Cor}(X_j, \tilde{X}_j) = 1 - s_j$.

- Choose $\Sigma_{\text{approx}} \approx \Sigma$, e.g., I or block-diagonal;
- $\hat{s} = \underset{s}{\text{Argmin}} \sum |1 - s_j|$ st $s \geq 0$, $\text{diag}(s) \preceq 2\Sigma_{\text{approx}}$;
- $\gamma = \underset{\gamma}{\text{Argmax}} \gamma$ st $\text{diag}(\gamma\hat{s}) \preceq 2\Sigma$;
- $s = \gamma\hat{s}$.

Use a statistic $W_j = w_j(X, \tilde{X}, y)$ such that

$$w_j((X, \tilde{X})_{\text{swap}(S)}, y) = \begin{cases} +w_j(X, \tilde{X}, y) & \text{if } j \notin S \\ -w_j(X, \tilde{X}, y) & \text{if } j \in S \end{cases}$$

for instance, a difference of lasso coefficients

$$Z_j = b_j(\lambda) \\ W_j = |Z_j| - |\tilde{Z}_j|$$

or

$$Z_j = \text{Max}\{\lambda : b_j(\lambda) \neq 0\} \\ W_j = \text{sign}(|Z_j| - |\tilde{Z}_j|) \text{Max}\{|Z_j|, |\tilde{Z}_j|\}.$$

Conditional on $|W_1|, \dots, |W_p|$, the signs of the null (i.e., irrelevant) W_j are iid coin flips. Estimate the false discovery proportion as

$$\widehat{\text{FDP}}(t) = \frac{\#\{j : W_j \leq -t\}}{\#\{j : W_j \geq t\}};$$

set the threshold to

$$\tau = \text{Min}\{t > 0 : \widehat{\text{FDP}}(t) \leq q\}$$

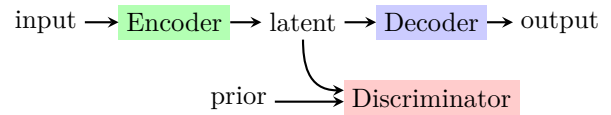
and select the variables $\hat{S} = \{j : W_j \geq \tau\}$: this controls the modified FDR

$$\text{mFDR} = \mathbb{E} \left[\frac{|\hat{S} \cap H_0|}{|\hat{S}| + 1/q} \right] \leq q$$

(add 1 to the $\widehat{\text{FDR}}$ numerator to control the FDR). R implementation: `knockoff`.

**Detection of accounting anomalies
in the latent space using
adversarial autoencoder neural nets**
M. Schreyer et al. (2019)

To add an arbitrary prior to the latent space of an autoencoder, e.g., a Gaussian mixture, use a discriminator.



**Learning sampling in financial statement
audits using VQ autoencoding neural networks**
M. Schreyer (2020)

The VQ-VAE

$$x \xrightarrow{q_\theta} z_e \mapsto z_q \xrightarrow{p_\phi} \tilde{x}$$

adds a “quantize” operation to the autoencoder, which maps the latent representation z_e to the closest vector z_q among a (learned) codebook e_ψ . The loss has 4 terms:

- Reconstruction from z_q [depends on ψ, ϕ];
- Reconstruction from z_e , skipping the reconstruction step $[\theta, \phi]$;
- $\|z_e - \tilde{z}_q\|$ to move the latent representation towards the codebook vectors $[\theta]$;
- $\|\tilde{z}_e - z_q\|$ to move the codebook vectors towards the latent representation $[\psi]$;

where $\bar{\cdot}$ is the *stop-gradient* operation: identity in the forward pass, zero in the backward pass – i.e., the quantity is considered constant.

**Counterexample-guided learning
of monotonic neural networks**
A. Sivaraman et al. (2020)

To ensure that a trained model is monotonic, compute its (upper and lower) *monotonic envelope*: $\bar{f}(x_0)$ comes from the maximum monotonic violation for $x \leq x_0$ and can be computed with an SMT solver – more precisely, an *OMT* (optimization modulo theories) solver such as OptiMathSAT proprietary.

One can also include those examples at training time, adding (to the next batch) counterexamples for each sample in the current batch, and assigning them the average label.

**Automatically learning compact quality aware
surrogates for optimization problems**
K. Wang et al. (2020)

In the *predict-then-optimize* framework

$$\xi \mapsto \theta \mapsto \underset{x}{\operatorname{Argmin}} f_{\theta}(x)$$

simplify the optimization layer by replacing $x \in \mathbf{R}^n$ with $x = Py$, $y \in \mathbf{R}^m$, $m \ll n$, where P is *learned* (rather than random).

[Implementation with `cvxpylayers` – `qpth` is older and limited to quadratic programming.]

**Generative minimization networks:
training GANs without competition**
P. Grnarova et al. (2021)

To find the Nash equilibria of a min-max game

$$\text{Player 1 : } \underset{u}{\operatorname{Min}} \underset{v}{\operatorname{Max}} M(u, v)$$

$$\text{Player 2 : } \underset{v}{\operatorname{Max}} \underset{u}{\operatorname{Min}} M(u, v)$$

the often-used gradient descent ascent (GDA) need not converge. Instead, minimize the *duality gap*

$$\operatorname{DG}(u, v) = \underset{v'}{\operatorname{Max}} M(u, v') - \underset{u'}{\operatorname{Min}} M(u', v).$$

For GANs,

$$M(u, v) = \underset{\substack{z_1 \sim \text{data} \\ z_2 \sim \text{noise}}}{\operatorname{E}} [\log D_v(z_1) + \log(1 + D_v G_u z_2)].$$

To estimate the duality gap, run k steps of gradient descent to find u_{worst} and v_{worst} .

**Ultra data-efficient GAN training: drawing
a lottery ticket first, then training it toughly**
T. Chen et al. (2021)

To train a GAN with little data:

- Look for a lottery ticket, with *iterative magnitude pruning* (train for t epochs and 100 samples, prune the smallest weights rewind);
- Train with adversarial augmentation: perturbations to the intermediate features.

**Bootstrap your own latent:
a new approach to self-supervised learning**
J.B. Grill (2020)

BYOL performs self-supervised learning (SSL) without negative pairs by training two networks:

- An online network, which predicts the target representation of an augmented version of the input;
- A target network, moving average of the online network.

**Exploring simple Siamese
representation learning**
X. Chen and K. He (2020)

To learn a latent representation, use simple siamese networks (SimSiam): feed two augmentations of the same image to two copies of the same network, keeping one frozen (“stop-gradient”) and use the latent representation of one to predict that of the frozen one (that is BYOL without the moving average).

**Understanding self-supervised learning
dynamics without contrastive pairs**
Y. Tian

In BYOL/SimSiam, the stop gradient and the predictor are important; the moving average, weight decay, predictor optimality, less so.

DirectPred does not train the predictor, but sets its weights using the PCA of its input.

Non-additive measures
V. Torra et al. (2014)

A **non-additive measure** (or *capacity*, or fuzzy measure) satisfies

$$A \subset B \Rightarrow \mu(A) \leq \mu(B).$$

It can model negative and positive interactions,

$$\begin{aligned} \mu(A \sqcup B) &< \mu(A) + \mu(B) \\ \mu(A \sqcup B) &> \mu(A) + \mu(B). \end{aligned}$$

For instance, a 2-book set $\{x_1, x_2\}$ may be more valuable than the individual books,

$$v(\{x_1, x_2\}) > v(\{x_1\}) + v(\{x_2\}).$$

In a workshop, if $\mu(A)$ is the number of widgets produced by the workers in A , we have $\mu(A \sqcup B) = \mu(A) + \mu(B)$ if they work independently, $\mu(A \sqcup B) > \mu(A) + \mu(B)$ if they cooperate, $\mu(A \sqcup B) < \mu(A) + \mu(B)$ if they interfere.

Examples include

- Sugeno λ -measures,

$$\mu(A \sqcup B) = \mu(A) + \mu(B) + \lambda\mu(A)\mu(B);$$

- Distorted probabilities $\mu(A) = f(P(A))$, with f non-decreasing and P a probability;
- \perp -decomposable non-additive measures, $\mu(A \sqcup B) = \mu(A) \perp \mu(B)$, where \perp is a t -conorm,

$$a \perp b = b \perp a$$

$$a \leq c, b \leq d \Rightarrow a \perp b \leq c \perp d$$

$$a \perp (b \perp c) = (a \perp b) \perp c$$

$$a \perp 0 = a$$

The *Möbius transform* of μ is

$$m(A) = \sum_{B \subset A} (-1)^{|A|-|B|} \mu(B)$$

and satisfies

$$\mu(A) = \sum_{B \subset A} m(B).$$

The **Choquet integral** is

$$\begin{aligned} C_\mu(f) &= \int_0^\infty \mu_f(r) dr \\ \mu_f(r) &= \mu\{x : f(x) > r\} \\ C_\mu(f) &= \sum (f(x_{s(i)}) - f(x_{s(i-1)})) \mu(A_{s(i)}) \\ A_{s(i)} &= \{x_{s(i)}, \dots, x_{s(N)}\} \end{aligned}$$

where the $f(x_{s(i)})$ are in increasing order. (In the book example, $C_v(n)$ is the value of n_1 copies of volume 1 and n_2 copies of volume 2, if we sell $n_1 \wedge n_2$ 2-book sets and the rest as individual volumes.) The *Sugeno integral* is

$$\begin{aligned} S_\mu(f) &= \sum_{r \in [0,1]} r \wedge \mu_f(r) \\ &= \text{Max}_i \text{Min}\{f(x_{s(i)}), \mu(A_{s(i)})\}. \end{aligned}$$

The *generalized fuzzy integral* of a simple function

$$f = \bigoplus_{i=1}^n a_i \mathbf{1}_{A_i} \quad A_1 \supseteq \dots \supseteq A_n$$

is

$$\text{GF}_\mu(f) = \bigoplus a_i \boxdot \mu(A_i).$$

The Choquet integral is obtained with $\oplus = +$ and $\boxdot = \cdot$, the Sugeno integral with $\oplus = \max$ and $\boxdot = \min$.

There are several notions of *entropy* for non-additive measures.

$$\begin{aligned} H_Y(v) &= \sum_i h\left(\sum_{\substack{A \subset N \\ i \notin A}} \gamma_{|A|}^n (v(A \cup \{i\}) - v(A))\right) \\ H_{MR}(v) &= \sum_i \sum_{\substack{A \subset N \\ i \notin A}} \gamma_{|A|}^n h(v(A \cup \{i\}) - v(A)) \\ h(x) &= -x \log x \\ \gamma_k^n &= \frac{(n-k-1)! k!}{n!}. \end{aligned}$$

A maximal chain $\emptyset = C_0 \subsetneq \dots \subsetneq C_n = \llbracket 1, n \rrbracket$ defines a probability distribution

$$p_C = (v(C_1) - v(C_0), \dots, v(C_n) - v(C_{n-1}))$$

and yet another entropy

$$M_{\min}(v) = \text{Min}_C H(p_C).$$

5a. The von Neumann-Morgenstern theorem states that a preference order on subsets of a convex set X satisfies

- (i) $p \succ q \Rightarrow \lambda p(1-\lambda)r \succ \lambda q + (1-\lambda)r$
- (ii) $p \succ q$ and $q \succ r \Rightarrow \exists \alpha, \beta \in (0, 1)$

$$\alpha p + (1-\alpha)r \prec q \prec \beta p + (1-\beta)r$$

iff there exists $u : \mathcal{P}(X) \rightarrow \mathbf{R}$ such that

- $p \succ q \Leftrightarrow u(p) \geq u(q)$
- (affine) $\forall p, q \in \mathcal{P}(X) \forall \lambda \in [0, 1]$

$$u(\lambda p + (1-\lambda)q) = \lambda u(p) + (1-\lambda)u(q)$$

Such a u is unique up to a positive affine transformation. This result can be generalized to *mixture spaces* (replace $\lambda p + (1-\lambda)q$ with $h_\lambda(p, q)$), *subjective probabilities*, i.e., preference orders such that

$$E \cap G = F \cap G = \emptyset \Rightarrow (E \succ F \Leftrightarrow E \cup G \succ F \cup G),$$

and *Savage acts*, i.e., maps $f : S \rightarrow X$.

5b. An urn contains 30 red balls and 60 black-or-yellow balls; you are presented with four lotteries:

- Get \$100 if you pick a red ball;
- Get \$100 if you pick a black ball;
- Get \$100 if you pick a red or yellow ball;
- Get \$100 if you pick a black or yellow ball;

For most people, $R \succ B$, but $RY \prec BY$: this cannot be formalized by expected utility theory (**Ellsberg's paradox**), but can be with the Choquet integral utility model, which accounts for both uncertainty (probability theory) and imprecision (or knightian uncertainty – fuzzy sets).

5c. The Choquet integral can be generalized to functions taking negative values,

$$\int a dv = \int_{-\infty}^0 [v(a \geq y) - v(S)] dy + \int_0^\infty v(a \geq y) dy.$$

If v is *convex* (supermodular), i.e.,

$$\forall A, B \quad v(A \cup B) + v(A \cap B) \geq v(A) + v(B),$$

its core $\{p : \forall A \ p(A) \geq v(A)\}$ is non-empty and

$$\int a dv = \text{Min}_{p \in \text{Core}(v)} \int a dp.$$

6. The *Möbius transform* can be defined for a function on a poset $f : (P, \leq) \rightarrow \mathbf{R}$.

$$\begin{aligned} \mu(x, y) &= \begin{cases} 1 & \text{if } x = y \\ -\sum_{y \leq z < x} \mu(y, z) & \text{if } y < x \\ 0 & \text{otherwise} \end{cases} \\ \Delta^f(x) &= \sum_{y \leq x} \mu(x, y) f(y) \\ f(x) &= \sum_{y \leq x} \Delta^f(y) \end{aligned}$$

It measures the extra contribution not achieved by smaller coalitions.

Let \bar{v} be the multilinear extension of the non-additive measure $v : \{0, 1\}^n \rightarrow \mathbf{R}$. The **Shapley value** is

$$\begin{aligned} \phi_i(v) &= \sum_{S \ni i} \frac{1}{|S|} \Delta^v(S) \\ &= \int_0^1 \frac{\partial}{\partial x_i} \bar{v}(t, \dots, t) dt. \end{aligned}$$

It is characterized by:

- Symmetry: $\phi_{\sigma(i)}(\sigma v) = \phi_i(v)$;
- Efficiency: $v(X) = \sum \phi_i(v)$;
- Null-zero: $(\forall S \ v(S \cup i) = v(S)) \Rightarrow \phi_i(v) = 0$;
- Additivity: $\phi(v + w) = \phi(v) + \phi(w)$.

It can be generalized

- From singletons $\{i\}$ to arbitrary subsets;
- To bi-cooperative games, where $v(S, T)$ is the value if voters in S vote +1, those in T vote -1, and the rest abstain;
- To a subset of $\mathcal{P}(X)$ (feasible coalitions), described by a graph.

***Understanding machine learning
for diversified portfolio construction
by explainable AI***
M. Jaeger et al. (2020)

Block-bootstrap the investment universe (16 assets, 20 years, daily, 1- or 3-month blocks), build HRP and ERC portfolios (monthly rebalancing), forecast Calmar(HRP) – Calmar(ERC) from bootstrap features (presence of trend, stability of this trend, dispersion, how good an approximation the hierarchy-filtered correlation matrix is, etc.) with xgboost, and explain with Shapley values.

The Shapley vs feature plot is an alternative to partial dependency plots.

Return attribution
C.R. Bacon and M.A. Wright (2012)

(Textbook material)

Performance attribution: history and progress
C.R. Bacon (2019)

(Review)

***Geometric attribution
and the interaction effect***
A.E. Weber (2018)

There are many variants of Brinson attribution: arithmetic or geometric, with or without interaction.

$$\begin{aligned} r &= \sum w_i w_i \\ \bar{r} &= \sum \bar{w}_i \bar{r}_i \\ A &= \sum w_i \bar{r}_i \\ S &= \sum \bar{w}_i r_i \end{aligned}$$

$$\begin{aligned} r - \bar{r} &= \sum (w_i - \bar{w}_i)(\bar{r}_i - \bar{r}) + \sum \bar{w}_i(r_i - \bar{r}_i) + \sum I_i \\ r = \bar{r} &= \sum (w_i - \bar{w}_i)(\bar{r}_i - \bar{r}) + \sum w_i(r_i - \bar{r}_i) \\ \frac{1+r}{1+\bar{r}} &= \frac{1+r_A}{1+\bar{r}} \cdot \frac{1+r}{1+r_A} \\ \frac{1+r}{1+\bar{r}} &= \frac{1+r_A}{1+\bar{r}} \cdot \frac{1+r}{1+r_S} \cdot \frac{(1+\bar{r})(1+r)}{(1+r_A)(1+r_S)} \end{aligned}$$

(For the geometric ones, write each factor as 1+contribution.)

Risk-adjusted performance attribution
J.D. Fisher and J. D'Alessandro (2018)

The CAPM $R_P = \alpha + \beta R_M = \alpha + R_B + (1 - \beta)R_B$ suggests adding a risk term to Brinson analysis (it is easier to outperform by taking on more risk):

$$R_P - R_B = \alpha + (1 - \beta)R_B.$$

Risk-adjusted performance attribution
D. Spaulding (2016)

Brinson analysis can be adjusted for risk: replace the returns μ with

$$MM = \frac{\sigma_{\text{Benchmark}}}{\sigma_{\text{Portfolio}}} \cdot \mu.$$

Attribution hears a Who!
A. Muralidhar (2016)

Replace MM with

$$M^3 = \sqrt{T} \frac{(\mu_P - \frac{1}{2}\sigma_P^2) - (\mu_B - \frac{1}{2}\sigma_B^2)}{\sqrt{\sigma_P^2 - \sigma_B^2 - 2\sigma_P\sigma_B\rho}}$$

where T is the number of years of data.

***Dynamic segment timing and the predictability
of actively managed mutual fund returns***
J.C. Hsu et al. (2016)

The Hsu-Kalesnik-Myers (HKM) model generalizes the multi-period Brinson model by decomposing the returns into static, dynamic and stock-specific components.

***Performance attribution:
measuring dynamic allocation skill***
J.C. Hsu et al. (2010)

HKM model

***Absolute return equity risk attribution
and forecasting***
R.A. Cooper and T. Li (2012)

Return attribution $\mu = \sum \mu_i$ defines a volatility and information ratio attribution:

$$\begin{aligned} \sigma &= \frac{\sigma^2}{\sigma} = \frac{\text{Var} \sum X_i}{\sigma} = \frac{\sum \text{Cov}(X_i, X)}{\sigma} = \sum \rho_i \sigma_i \\ IR &= \frac{\mu}{\sigma} = \frac{\sum \mu_i}{\sigma} = \sum \frac{\rho_i \sigma_i}{\sigma} \cdot \frac{\mu_i}{\rho_i \sigma_i} = \sum \rho_i \frac{\sigma_i}{\sigma} IR_i \end{aligned}$$

Effective return: a breakthrough in cumulative performance attribution
R. Surz (2010)

For multi-period return attribution:

- Replace the time-changing returns r_{ijt} with constant returns r_{ij} leading to the same cumulated returns;
- weak them ever so slightly so that r_i , \bar{r}_i and \bar{r} have the correct values;
- Then, use 1-period Brinson analysis.

Conditional benchmarks and predictors of mutual fund performance
S. Cederburg (2018)

Managerial skill can be measured from the residuals of a Cahart 4-factor regression. Allow non-constant regression coefficients by estimating them (with the generalized method of moments, GMM) from some covariate, e.g., the 3-month lagged factor loadings (computed from portfolio weights and (constant) constituent factor exposure).

[One could also use a Kalman filter.]

Non-linear factor attribution
S. De Boer (2019)

B : exposures

$$X = Bf + \varepsilon$$

P : factor-mimicking portfolios

$$w = Bp + \eta$$

η : residual weights

$$\omega = \frac{\Omega\eta}{\eta'\Omega\eta}$$

$$C_{ik} = \frac{\omega_{ik}^2}{\sum_{\ell} \omega_{i\ell}^2}$$

$$\Delta = \varepsilon' \text{diag}(\eta)C$$

Characteristics-based factors
Z. Chen et al. (2018)

There is a difference between stock characteristics (e.g., momentum) and stock exposures (e.g., beta to a momentum portfolio): only the former explain returns – using the latter to adjust returns (residuals of a regression against the factor portfolio) is a bad idea.

Arbitrage portfolios
Z. Chen et al. (2018)

Look at double-sorted portfolios, wrt both stock characteristics and beta (for this characteristic).

Smart beta multifactor methodology: mixing vs integrating
T. Chow et al. (2017)

To combine investment factors, one could invest in stocks with a good overall score, or combine single factor portfolios – the former has better performance, but higher turnover and concentration.

Downsampling time series for visual representation
S. Steinarsson (2013)

Split the data into (equal-size) buckets, with the first and last points alone in their bucket.

- Keeping only the median (or the mode) in each bucket tends to remove local extrema.
- The longest line algorithm picks a point in each bucket to maximize the length of the resulting line (dynamic programming).
- The **Douglas-Peucker** algorithm picks the point farthest away from the line from the first to the last point, and iterates until the points are sufficiently close.
- The *Visvalingam-Whyatt* algorithm measures the importance of a point as the area of the triangle it forms with its adjacent points.
- The *largest triangle* algorithm picks the point in each bucket with the largest area with the point selected in the previous bucket and the average point in the next one.

We can use dynamic bucket sizes (to keep outliers):

- Start with equal-sized buckets;
- Fit a linear regression in each bucket (also include one point before and one point after);
- Split the bucket with the largest residuals, and merge the pair with the smallest residuals.

A Bayesian graphical approach for large-scale portfolio management with fewer historical data
S. Oya (2021)

The L^1 penalty in the graphical lasso can be seen as a Laplace prior; the posterior can be sampled with Gibbs sampling (positive definiteness complicates matters).

The Shapley decomposition for portfolio risk
S. Mussard and V. Terraza (2008)

The variance of a portfolio is $w'Vw = \sum w_i\sigma_i^2 + 2\sum_{i<j} w_iw_j\rho_{ij}\sigma_i\sigma_j$; the second term, the “between security risk”, can be decomposed into individual asset contributions with Shapley values.

Portfolio performance attribution via Shapley value
N. Moehle et al. (2021)

Use Shapley values to decompose any performance measure into the contribution of features that can be turned on or off, e.g., inputs (used or not), constraints, terms in the optimization objective, etc.

Variance allocation and Shapley value

R. Colini-Baldeschi et al. (2017)

The Shapley decomposition of a variance game is

$$\text{Var}\left[\sum X_i\right] = \sum_i \text{Cov}\left(X_i, \sum_j X_j\right).$$

The Shapley value decomposition of optimal portfolios

H. Shalit (2017)

Use Shapley values to decompose the risk (volatility) of the mean-variance and the *mean-Gini* portfolio built on a set of assets.

The *Gini mean difference* is

$$\text{GMD}(X) = \frac{1}{2} \mathbb{E}_{X_1, X_2} |X_1 - X_2| = 2 \text{Cov}(X, F_X(X)).$$

Deep learning for portfolio

Z. Zhang et al. (2020)

LSTM on prices, for 4 assets (ETFs on equities, bonds, commodities and VIX), to compute portfolio weights (softmax) maximizing the Sharpe ratio.

Portfolio construction as linearly constrained separable optimization

N. Moehle et al. (2021)

The portfolio optimization problem

$$\begin{array}{ll} \text{Find} & h \\ \text{To minimize} & \alpha'h - \gamma h'Vh - \phi(h) \\ \text{Such that} & \ell \leq \mathbf{1}'h \leq u \\ \text{Where} & V = X\Sigma X' + D \end{array}$$

where ϕ models transaction costs, minimum trade size, tax liability, position limits, minimum holdings size, integer share constraints, can be formulated as a *separable affine problem*

$$\begin{array}{ll} \text{Find} & h, y \in \mathbf{R}^n, c \in \mathbf{R} \\ \text{To maximize} & \alpha'h - \gamma(y'y + h'Dh) \\ \text{Such that} & y = CXh \\ & c + \mathbf{1}'h = 1 \\ & 1 - u \leq c \leq 1 - \ell \\ \text{Where} & \Sigma = CC' \text{ Cholesky} \end{array}$$

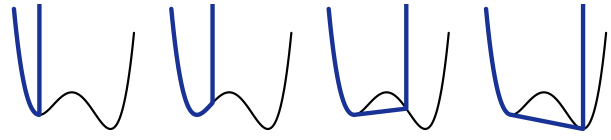
i.e., of the form (after introducing slack variables to remove the inequalities)

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & \sum f(x_i) \\ \text{Such that} & Ax = b. \end{array}$$

For a lower bound, solve the convex problem

$$\text{Minimize}_x \sum f_i^{**}(x_i) \text{ such that } Ax = b$$

where f^{**} is the *convex envelope* of f .



The problem

$$\begin{array}{ll} \text{Find} & x, z \in \mathbf{R}^n \\ \text{To minimize} & I_A(z) + \sum f_i(x_i) \\ \text{Such that} & x = z \\ \text{Where} & I_A(z) = \begin{cases} 0 & \text{if } Az = b \\ \infty & \text{otherwise} \end{cases} \end{array}$$

can be solved with ADMM:

$$\begin{array}{l} L(x, z, \lambda) = f(x) + I_A(z) + \frac{1}{2} \|x - z + \lambda\|^2 \\ x \leftarrow \underset{x}{\text{Argmin}} L(x, z, \lambda) \\ z \leftarrow \underset{z}{\text{Argmin}} L(x, z, \lambda) \\ \lambda \leftarrow \lambda + x - z. \end{array}$$

The z update reduces to solving a linear system, and the x updates are univariate $x_i \leftarrow \text{prox}_{f_i}(z_i - \lambda_i)$, with f_i piecewise quadratic. ADMM is guaranteed to converge for convex problems: since it is non-convex, initialize with the relaxed problem.

Scaling may help

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & \sum f_i(x_i) \\ \text{Such that} & Ax = b \end{array} \rightsquigarrow \begin{array}{ll} \text{Find} & \tilde{x} \\ \text{To minimize} & \sum f_i(E_{ii}\tilde{x}_i) \\ \text{Such that} & DAE\tilde{x} = Db \\ \text{Where} & \tilde{x} = E^{-1}x \\ & E \text{ diagonal} \\ & D \text{ invertible} \end{array}$$

Exploring the factor zoo with a machine-learning portfolio

H. Sak et al. (2019)

Train an ensemble of models, for 100 stocks, on a moving window; only use predictors with a significant Fama-French alpha in the training window.

Deep fundamental factor models

M.F. Dixon and N.G. Polson (2020)

Fit a neural network on a moving window, to predict the returns of 3000 assets from 50 factors. Use the gradient and the Hessian (of the output wrt the input) to find the most important factors and interactions.

Deep factor model
K. Nakagawa et al. (2018)

Deep factor models (17 factors to predict the returns of 2000 Topix stocks with a fully-connected neural net) can be interpreted with *layerwise relevance propagation* (LRP) [yes, this ignores the nonlinearity].

$$\begin{aligned} \ell &: \text{layer} \\ i, j &: \text{neurons} \\ x &: \text{activations (before non-linearity)} \\ z &: \text{activations (after)} \\ g &: \text{non-linearity} \\ R_i^\ell &: \text{contribution of neuron } i \\ R^n &: \text{output} \\ R_i^\ell &= \sum_j \frac{z_{ij}}{z_{\bullet j}} R_j^{\ell+1} \\ z_{ij} &= w_{ij}^\ell x_i^\ell \\ x_i^{\ell+1} &= g(z_{i\bullet}) = g\left(\sum s_{ij}^\ell x_i^\ell\right) \end{aligned}$$

To improve numeric stability, replace $z_{ij}/z_{\bullet j}$ with

$$\frac{z_{ij}}{z_{\bullet j} + \varepsilon \text{sign}(z_{\bullet j})} \text{ or } \frac{z_{ij}^+}{z_{\bullet j}^+} - \frac{z_{ij}^-}{z_{\bullet j}^-}$$

**Deep recurrent factor model:
interpretable non-linear
and time-varying multi-factor model**
K. Nakagawa et al. (2019)

Layerwise relevance propagation to interpret LSTM-based factor models (500 stocks, 17 factors); the contributions are comparable to those of a linear model.

**Interpreting deep learning models
with marginal attribution
by conditioning on quantiles**
M. Merz et al. (2021)

The first- and second-order *marginal attribution by conditioning on quantiles* (MACQ) of a model $x \mapsto \mu(x)$ are

$$\begin{aligned} S_j &= \mathbb{E} \left[X_j \frac{\partial \mu(X)}{\partial X_j} \middle| \mu(X) = \mu_\alpha \right] \\ T_{ij} &= \mathbb{E} \left[X_i X_j \frac{\partial^2 \mu(X)}{\partial X_i \partial X_j} \middle| \mu(X) = \mu_\alpha \right] \end{aligned}$$

where μ_α is the α quantile of $\mu(X)$. If the origin 0 is chosen in the middle of the data,

$$\mu_\alpha \approx \mu(0) + \sum_i S_i - \frac{1}{2} \sum_{ij} T_{ij}.$$

Monotone and partially monotone neural nets
H. Daniels and M. Velikova (2010)

Neural networks with just one hidden layer cannot approximate arbitrary monotonic functions, but with two, they can, for instance, with a *min-max network* (but it will not work well in practice).

$$x \mapsto \min_i \max_j \sigma \left(\sum w_{ijk} x_k \right)$$

Certified monotonic neural networks
X. Liu et al. (2020)

Add a penalty

$$\mathbb{E}_{x \sim \text{Unif}} \sum_i \text{Max} \{b, -\partial_{x_i} f(x)\}^2$$

to ensure that the derivatives of f are at least $b \geq 0$.

Proving that a ReLU network is monotonic can be formulated as a mixed integer program. To speed up computations, enforce and test monotonicity for pairs of consecutive layers (instead of the whole network).

One can also look for monotonicity violations around a given observation.

**A biclustering method
for time series data analysis**
J. Lee et al. (2009)

The **plaid model** for biclustering is

$$Z_{ij} = \mu_0 + \sum_k (\mu_k + \alpha_{ik} + \beta_{kj}) \rho_{ik} \kappa_{kj} + \varepsilon_{ij},$$

where ρ, κ are binary variables indicating cluster membership (of cluster k , for row i and column j), and $\mu_k + \alpha_{ik} + \beta_{kj}$ describes cluster k . The *layers* (biclusters) can be extracted one by one, by alternately estimating and (ρ, κ) .

For time series, ensure that the clusters in the time dimension are convex by filling in gaps of size at most C , and only keeping the largest convex component (there is also a pruning step, limited to the interval extremities in the time dimension).

**Improved biclustering
of microarray data demonstrated through
systematic performance tests**
H. Turner et al. (2005)

The original plaid algorithm used alternating least squares (3 steps: ρ , κ and $\mu + \alpha + \beta$); instead, use *binary least squares*

$$\rho = \text{Argmin}_{\rho \in \{0,1\}^n} \sum_{ij} (z_{ij} - \rho_i x_j)^2$$

(the objective is separable).

Plaid models for gene expression data
L. Lazzeroni and A. Owen (2002)

(First paper on the plaid model.)

Biclustering of expression data
Y. Cheng and G.M. Church (2000)

To find a δ -bicluster, *i.e.*, a short subset of rows I and columns J such that the *mean square residue*

$$H(I, J) = \text{Mean}_{i \in I} (a_{ij} - a_{iJ} - a_{IJ} + a_{IJ})^2$$

be below δ , one could start with the full matrix and greedily remove the row or column reducing $H(I, J)$ the most, until we reach δ (or no row or column reduces H). Instead,

- Remove the rows (resp. columns) with

$$d(i) = \text{Mean}_{j \in J} (a_{ij} - a_{iJ} - a_{IJ} + a_{IJ})^2 > \alpha H(I, J);$$

- Remove the rows (columns) with the largest $d(i)$;
- Add back rows (columns) with $d(i) \geq H(I, J)$.

(Also look at the row variance

$$V(I, J) = \text{Mean}_{j \in J} (a_{ij} - a_{IJ})^2$$

to reject trivial clusters.)

The Gibbs-plaid biclustering model
T. Chekouo et al. (2015)

Plaid model with a prior on the gene labels ρ_{ik} and the condition labels κ_{jk} , from similarities from the gene ontology (GO) annotations and proximity in time respectively.

Extracting conserved gene expression motifs from gene expression data
T.M. Murali and S. Kasif (2003)

A conserved motif (*xMotif*) is a pair (I, J) of genes and samples such that $\forall i \in I \quad \forall j \in J \quad a_i \leq x_j b_i$. To find them:

- Pick k samples at random (seed);
- Find the genes on which they agree;
- Find the samples on which those genes agree;
- Discard if unsatisfactory;
- Start again.

Spectral biclustering of microarray data: coclustering genes and conditions
Y. Kuger et al. (2003)

SVD to uncover checkerboard patterns.

QuBic: a qualitative biclustering algorithm for gene expression data
G. Lie et al. (2009)

The QuBic qualitative clustering algorithm

- Starts with a graph, with genes as vertices, and gene similarity as edge weights;
- Picks the heaviest edge;
- Builds a cluster around it, progressively adding genes, such that the proportion of samples with the same gene expression (terciles) remains above 95%.

A systematic comparison and evaluation of clustering methods for gene expression data
A. Prelić et al. (2006)

The *bimax* algorithm, for binary data:

- Picks a row as a template, and divides the columns into two groups: 1s in the template, and 0s in the template;
- Divides the rows into three groups:
 - With 0s where there are 0s in the template;
 - With 0s where there are 1s in the template;
 - The rest;
- Proceeds recursively (the two blocks of zeroes can be ignored).

A linear time biclustering algorithm for time series gene expression data
S.C. Madeira and A.L. Oliveira (2005)

Discretize the data and look for biclusters with the same value in each column (date), from the *suffix tree* (Ukkonen) of the rows.

```

N1 U2 D3 U4 N5
D1 U2 D3 U4 D5
N1 N2 N3 U4 N5
U1 U2 D3 U4 U5

```

Biclustering gene expression using factor graphs and the max-sum algorithm
M. Denitto et al. (2015)

The biclustering problem can be solved iteratively, by looking for the largest bicluster, replacing with with noise, and iterating. The problem of finding the largest bicluster can be modeled with a *factor graph* and binary variables x_{ij} for cluster membership:

- $a_{ij}x_{ij}$ to promote high-value clusters (high gene expression);
- $d(a_{ij}, a_{kl})x_{ij}x_{kl}$ to promote coherent clusters;
- A factor for each pair of rows i, k (resp. columns), zero if either is zero or if they have the same (column) pattern

$$I \left\{ \sum_j x_{ij} = 0 \text{ or } \sum_\ell x_{k\ell} = 0 \text{ or } \sum_j (x_{ij} - x_{kj})^2 = 0 \right\}$$

to get submatrices

(Prior approaches used exemplars and had a quartic number of factors.) The problem can be efficiently solved with the MaxSum message passing algorithm (the messages can be computed explicitly).

A biclustering approach based on factor graphs and the max-sum algorithm
M. Denitto et al. (2017)

More detailed version of the same paper.

**Biclustering of time series data
using factor graphs**
M. Denitto et al. (2017)

**HOP-MAP: efficient message passing
with high-order potentials**
D. Tarlow et al. (2010)

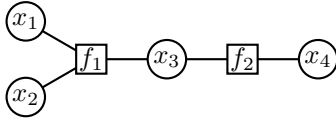
In a factor graph (with binary variables), with high-order potentials such as

- Cardinality: $\theta(h_1, \dots, h_N) = f(h_1 + \dots + h_N)$,
- Convexity: $\theta(h_1, \dots, h_N) = \mathbf{1}_{\{i : h_i=1\} \text{ convex}}$,
- Ordering: $\theta(h_x, h_y) = 1$ if exactly one coordinate, i , of h_x is 1, exactly one coordinate, j , of h_y , is 1, and $i < j$,

the messages can be computed efficiently with dynamic programming.

**Factor graphs, message passing,
loopy belief propagation**

A **factor graph** is a factorization of an (unnormalized) probability distribution as $p(x) \propto \prod_f \phi_f(x_f)$, where each ϕ_f only depends on a subset x_f of the variables x_1, \dots, x_N . It can be represented as a bipartite graph, with nodes for variables x_i and factors ϕ_f , and edges whenever a variable is used in a factor.



$$p(x_1, x_2, x_3, x_4) \propto \phi(x_1, x_2, x_3) \phi(x_3, x_4)$$

In the case of trees, the probability can be maximized by *message passing*, i.e., exchanging messages between nodes (each message to/from a variable is a function of its possible values – for a discrete variable with k values, it is a dimension- k vector).

$$\mu_{x \rightarrow f}(x) = 0$$

$$\mu_{f \rightarrow x}(x) = \log f(x)$$

$$\mu_{f \rightarrow x}(x) = \max_{x_1, \dots, x_M} \log f(x, x_1, \dots, x_M) + \sum_m \mu_{x_m \rightarrow f}(x_m)$$

$$\mu_{x \rightarrow f}(x) = \sum_{\ell \in \mathcal{N}(x)} \mu_{f_\ell \rightarrow x}(x).$$

Empirically, the **max-sum** algorithm still gives good results for non-trees (loopy belief propagation).

**Simultaneous supervised clustering
and feature selection over a graph**
X. Shen et al. (2012)

The *fused lasso*, a penalty on $\sum |\beta_{i+1} - \beta_i|$, clusters the predictors of a linear regression when there is an ordering on them. This ordering can be represented by a graph $\bullet \cdots \bullet$: the fused lasso can be generalized to an arbitrary (e.g., complete) graph. The L^1 norm in the penalties $\sum |\beta_i|$ (for irrelevant predictors) and

$\sum |\beta_i - \beta_j|$ can be replaced with $\text{Min}\{|\beta|, 1\}$: the objective is no longer convex, but a difference of convex functions, amenable to *difference convex programming* (or minimization-majorization, MM).

$$\frac{\text{Min}\{|\beta|, 1\}}{|\beta|} = \frac{1}{|\beta|} - \frac{1}{(|\beta|-1)_+}$$

Derivative analytics with Python
Y. Hilpisch (2015)

2. An option-pricing framework should account for different sources of risk: level, volatility, jump, and interest rate.

Perfect hedging is only possible with continuous rebalancing and in complete markets – in particular, without jumps.

3. For options, market stylized facts include:

- Returns have fat tails, prices present jumps;
- Volatility is stochastic, tends to cluster, mean-reverts, is negatively correlated with returns (leverage);
- Implied volatility has a smile, which is flatter for longer-term options (term structure);
- Interest rates are stochastic, mean-reverting, and often positive.

A market model should include stochastic volatility, jumps and stochastic short rates.

4. There is no arbitrage (in continuous time, “no free lunch with vanishing risk”) iff there is an equivalent martingale measure Q . It is unique iff the market is complete.

6a. If the risk-neutral density (RND) q is known, the price of a European option with payoff C_T can be computed as

$$C_0 = e^{-rT} \int_0^\infty C_T(s) q(s) ds.$$

The characteristic function (Fourier transform) $\hat{q}(u) = \mathbb{E}_{X \sim q}[e^{iuX}]$ is often known: the option price can be computed using Parseval’s identity (I ignore signs, normalization factors, discounting factor),

$$C_0 = \langle C_T, q \rangle = \langle \hat{C}_T, \hat{q} \rangle,$$

leading to the **Lewis formula**

$$C_0 = S_0 - \frac{\sqrt{S_0 K} e^{-rT/2}}{\pi} \int_0^\infty \text{Re}[e^{-izk} \phi(-z-i/2)] \frac{dz}{z^2 + \frac{1}{4}}.$$

6b. The **Carr-Madan formulas** compute \hat{C}_0 (where C_0 is a function of $k = \log K$) using Fubini’s theorem (simple closed-form formula); the option price can then be computed with the (inverse) fast-Fourier transform (FFT).

7. For American options, the option value V and its continuation value C can be computed as (primal ap-

proach)

$$\begin{aligned} V_t &= \sup_{\tau} E_0^Q [B_0(\tau) h_{\tau}(S_{\tau})] \\ C_t &= E_t^Q [e^{-r\Delta t} V_{t+\Delta t}(S_{t+\Delta t}) | S_t = s] \\ V_t(s) &= \text{Max}\{h_t(s), C_t(s)\}. \end{aligned}$$

This is a lower bound on the option value. The dual approach gives an upper bound.

$$\begin{aligned} V_0 &= \sum_{\tau} E_0^Q [B_0(\tau) h_{\tau}(S_{\tau})] \\ &= E_0^Q \text{Max}_t B_0(t) (h_t(S_t) - Q_t) \\ Q_t &= \sum_{u \leq t} (V_u(S_u) - E_{u-\Delta t}^0 V_u(S_u)). \end{aligned}$$

Both rely on least squares Monte Carlo (LSM, Longstaff-Schwartz).

8. The following models extend the Black-Scholes model (BSM73):

- Merton (M76): jumps;
- Heston (H93): stochastic volatility;
- Bates (B96): jumps and stochastic volatility;
- BCC97: jumps, stochastic volatility and stochastic rates.

$$\begin{aligned} \frac{dS}{S} &= (r - r_J)dt + v_t dZ_1 + JdN \\ dv &= \kappa(\theta - v)dt + \sigma\sqrt{v}dZ_2 \\ dr &= \kappa(\theta - r)dt + \sigma\sqrt{r}dZ_3 \quad (\text{CIR}) \end{aligned}$$

For a single CIR process, one can explicitly compute $B_0(T)$.

The characteristic function of the BCC97 model is the product of those of H93 and M76J (the stochastic rate only intervenes through $B_0(T)$).

Option valuation can be done by solving a PDE (the formula is similar to the Black-Scholes one, but involves an integral), Fourier transform (Lewis or Carr-Madan) or (for arbitrary options) Monte Carlo simulations (replace the drift with r , carefully discretize the SDE, simulate, and compute the average discounted payoff).

10. For Monte Carlo simulations, the price can be discretized with the Euler or log-Euler scheme. For the volatility, naive discretization is not exact: it can be discretized exactly, but this involves a non-central χ^2 distribution, which is slow; instead, one can truncate (or reflect – there are many variants) the naive discretization; this also easily accounts for the correlation between the volatility and price innovations.

Use all the Monte Carlo variance reduction tricks you can think of: antithetic variables, moment matching, control variates (European option prices, which can be computed exactly, when pricing Americal options).

11. The Black-Scholes model is complete; the Heston model (stochastic volatility) is not, but any option can be replicated with the underlying and one option.

The Bates model (with jumps) is not complete: one would need an infinite number of options to replicate any payoff. Consequently, the martingale measure Q is not unique: several prices are compatible with the absence of arbitrage. However, market prices correspond to one of those measures: *calibration* estimates it.

Calibrate the model parameters to liquid vanilla options (e.g., EURO STOXX 50, 3 maturities less than 1 year, 5 strikes, Eonia and Euribor up to 1 year), using, as loss function, the MSE of prices (for pricing), relative prices, or implied volatilities (for hedging), perhaps with some weights (e.g., vega-weighted implied volatility).

The CIR model can be calibrated from the forward rates; the forward rates can be computed from the yields

$$Y_T = -\frac{\log B_0(T)}{T} \quad f_T = Y_T + \frac{\partial Y_T}{\partial T} T.$$

Optimize in several steps: r , σ , J , and finally $\sigma + J$. There may be multiple local optimal: add an L^2 penalty to stay close to the previous-day solution.

13. Delta hedging is only exact if it is continuous, if the underlying is the only source of risk, if the price paths are continuous, and if volatility is constant.

Jump risk cannot be hedged away but, if there are options on many underlyers, it can be diversified away.

For a single option, delta hedging is insufficient: try to add options to the replicating portfolio.

Estimation of risk-neutral densities using positive convolution approximation **O. Bondarenko (2003)**

The risk-neutral density (RND) f satisfies

$$\text{Option price} = \int_0^{+\infty} \text{Payoff}(F) f(F) dF$$

and can be estimated from puts and calls

$$f(F) = \frac{\partial^2 P(k)}{\partial K^2} \Big|_{K=F} = \frac{\partial^2 C(k)}{\partial K^2} \Big|_{K=F}.$$

Intuitively, $f(x)dx$ is the price of an option paying \$1 if the spot S_T is in $[x, x + dx]$ and zero otherwise.

The *positive convolution approximation* (PCA) looks for a density of the form $f = \phi_h * u$, where ϕ_h is a fixed kernel (its shape is not that important, but its bandwidth is) and u an arbitrary density, to recover the put prices.

$$\begin{aligned} \text{Find} & \quad u \\ \text{To minimize} & \quad \sum [P_i - D^2 f(K_i)]^2 \\ \text{Such that} & \quad u \geq 0, \quad \int u = 1 \\ \text{Where} & \quad f = \phi_h * u \\ & \quad D^{-2} f(x) = \int_{-\infty}^x \int_{-\infty}^y f(z) dz dy \end{aligned}$$

After discretization, $f(x) = \sum a_j \phi(x - z_j)$, $\Delta z = 0.5h$, this becomes a quadratic optimization problem.

***A simple and reliable way to compute
option-based risk neutral distributions***
A.M. Malz (2014)

To estimate the risk-neutral density (RND) from the Breeden-Litzenberger formula

$$f(F) = \frac{\partial^2 P(k)}{\partial K^2} \Big|_{K=F} = \frac{\partial^2 C(k)}{\partial K^2} \Big|_{K=F}$$

fit the implied volatilities with a clamped cubic splines (set the first derivative to zero at the boundaries) and differentiate the Black-Scholes call prices. Check for no-arbitrage violations: the call price should be decreasing and convex.

***Exploring return dynamics
via corridor-implied volatility***
T.G. Andersen et al. (2015)

If there are no jumps, the realized variance is

$$E_0^Q[RV] = 2 \int_0^\infty \frac{M(K)}{K^2} dK$$

where $M(K) = \min\{P(K), C(K)\}$ and the realized variance is computed from the squared ratio- or log-returns (in the absence of jumps, they give the same result). With jumps, the result remains if we replace the squared returns with the difference

$$2(\text{ratio-returns} - \text{log-returns})$$

(which is approximately 2/3 of the squared ratio-returns and 1/3 of the log-returns).

The CBOE VIX estimates the *market-free implied volatility* (MFIV)

$$\text{MFIV} = \frac{2e^{rT}}{T} \left[\int_0^F \frac{P(K)}{K^2} dK + \int_F^\infty \frac{C(K)}{K^2} dK \right]$$

and is computed as

$$\begin{aligned} \sigma^2 &= \frac{2e^{rT}}{T} \sum \frac{\Delta K_i}{K_i^2} M(K_i) - \frac{1}{T} \left(\frac{F}{K_f} - 1 \right)^2 \\ \Delta K_i &= \frac{1}{2}(K_{i+1} - K_{i-1}) \\ \Delta K_1 &= K_2 - K_1 \\ \Delta K_N &= K_N - K_{N-1} \\ K_f &= \text{first strike below } F \end{aligned}$$

where only liquid strikes are used – but the corridor of strikes used, $[K_1, K_N]$, can change and create spurious jumps.

The *corridor implied volatility* uses the same formula for a more stable corridor, defined from the [3%, 97%] quantile range of the distribution defined by the cdf

$$R(K) = \frac{P(K)}{P(K) + C(K)}.$$

***Construction and interpretation
of model-free implied volatility***
T.G. Andersen and O. Bondarenko (2007)

Earlier paper on the corridor implied volatility (CIV), where the corridor is defined from the PCA (positive convolution approximation) risk-neutral density estimator.

There is a negative variance risk premium in the VIX.

***Model-free implied volatility
and its information content***
G.J. Jiang and Y.S. Tian (2003)

Derivation of the MFIV estimator from the Britten-Jones-Neuberger formula

$$E_0^Q \left[\int_{T_1}^{T_2} \left(\frac{dS_t}{S_t} \right)^2 \right] = 2 \int_0^\infty \frac{C(T_2, K) - C(T_1, K)}{K^2} dK$$

by adding a risk-free rate, setting $T_1 = 0$ so that the second term is a call payoff, either zero, or leading to a put price via the put-call parity.

***Option prices, implied price processes
and stochastic volatility***
M. Britten-Jones and A. Neuberger (2000)

(The formula is derived as the limit of a discrete model.)

Towards a theory of volatility trading
P. Carr and D. Madan (2002)

The model-free implied volatility (MFIV) can be computed as follows.

$$\begin{aligned} f(y) &= f(x) + f'(x)[(y-x)_+ + (x-y)_+] \\ &\quad + \int_0^x f''(t)(t-y)_+ dt + \int_x^\infty f''(t)(y-t)_+ dt \\ E_0^Q[f(F)] &= f(K)B_0 + f'(K)[C(K) - P(K)] \\ &\quad + \int_0^K f''(k)P(k)dk + \int_K^\infty f''(k)C(k)dk \end{aligned}$$

Letting $f(F) = \log(F_T/F_0)$ and $K = F_0$,

$$\begin{aligned} E_0^Q \left[\log \frac{F_T}{F_0} \right] &= - \int_0^{F_0} \frac{P(k)}{k^2} dk - \int_{F_0}^\infty \frac{C(k)}{k^2} dk \\ E_0^Q \left[\log \frac{F_T}{F_0} \right] &\stackrel{\text{It\^o}}{=} -\frac{1}{2} E_0^Q \int_0^T \sigma_t^2 dt. \end{aligned}$$

***Prices of state contingent claims
implicit in option prices***
D. Breeden and R. Litzenberger (1978)

Notice that

$$\frac{d}{dx} \int_x^\infty f(x, y) dy = \int_x^\infty \frac{\partial f(x, y)}{\partial x} dy - f(x, x)$$

and apply it to the price of a call option

$$C(K) = \int_0^\infty (S - K)_+ f(S) dS = \int_K^\infty (S - K) f(S) dS;$$

this gives

$$\frac{dC(K)}{dK} = \int_K^\infty -f(S) dS = 0$$

and finally

$$\frac{d^2 C(K)}{dK^2} = f(K).$$

***Is implied correlation worth calculating?
Evidence from foreign exchange options
and historical data***
C. Walter and J.A. Lopez (2000)

The correlation between (the log-returns of) two currency pairs A/C and B/C satisfies

$$\rho_{A,B/C} = \frac{\sigma_{A/C}^2 + \sigma_{B/C}^2 - \sigma_{A/B}^2}{2\sigma_{A/C}\sigma_{B/C}}.$$

Computing it from implied volatilities has sometimes (not always) some predictive power on future correlation.

***Implied correlation
for pricing multi-FX options***
P.V. Shechvhenko (2004)

For unrelated currency pairs, A/B and C/D,

$$\rho_{A/B,C/D} = \frac{\sigma_{C/B}^2 + \sigma_{A/D}^2 - \sigma_{D/B}^2 - \sigma_{C/A}^2}{2\sigma_{A/B}^2\sigma_{C/D}^2}.$$

***Implied correlation:
a new measure of diversification***
V.D. Skintzi and A.P.N. Refenes (2005)

The implied correlation is defined from the implied volatilities of an index and its components as

$$\rho = \frac{\sigma^2 - \sum_{i \neq j} w_i^2 \sigma_i^2}{\sum_{i \neq j} w_i w_j \sigma_i \sigma_j}.$$

***Measuring equity risk
with option implied correlations***
A. Buss and G. Vilkov (2012)

Adjust the historical correlations

$$\rho_{ij} = (1 - \lambda)\hat{\rho}_{ij} + \lambda$$

such that $\sigma^2 = \sum w_i w_j \sigma_i \sigma_j \hat{\rho}_{ij}$.

$$\lambda = \frac{\sigma^2 - \sum w_i w_j \sigma_i \sigma_j \hat{\rho}_{ij}}{\sum w_i w_j \sigma_i \sigma_j (1 - \hat{\rho}_{ij})}.$$

We can then define option implied betas

$$\beta_i = \frac{\sigma_i \sum w_j \sigma_j \rho_{ij}}{\sigma^2}.$$

***Estimating realistic implied
correlation matrix from option prices***
K.N. Numpacharoen (2013)

If $\sigma^2 > \hat{\sigma}^2$, replace $\rho_{ij} = (1 - \lambda)\hat{\rho}_{ij} + \lambda$ with

$$\rho_{ij} = (1 - \lambda)\hat{\rho}_{ij} - \frac{\lambda}{n - 1}.$$

Machine learning fund characterizations
D. Mehta et al. (2020)

Approaches to cluster funds (with k -means, hclust, fuzzy-C-means, SOM) often uses performance and risk measures as features (performance ratios, betas). Some use the bipartite graph of the top 10 holdings, projected, with spectral clustering.

Use aggregate data (proportion of assets in a given asset class, country, sector, capitalization bin, etc.) with a decision tree, random forest, or a neural net (FC, 3 layers, 512/256/128/ReLU, softmax) to reproduce the Morningstar classification (not clustering, but classification).

***Sequential deep learning for credit risk
monitoring with tabular financial data***
J.M. Clements et al. (2020)

Gradient boosted trees, commonly used to assess credit risk (from credit card data) do not leverage the time dimension of the data (without clever feature engineering): try temporal convolutions (TCN). Other baselines include MLP, TabNet (attention for tabular data) and LSTMs.

Winsorize using the boosted tree cutoffs.

***Covid-19 spreading in financial networks:
a semiparametric matrix regression model***
M. Billio et al. (2021)

Model the time-varying adjacency matrix of the financial network as

$$A_t = \sum_r B_r f_{rt} + E_t$$

$$E_t \sim \text{MN}(0, I, \Sigma)$$

$$\Sigma \text{ diagonal}$$

$$A_t, B_r, E_t \text{ matrices}$$

$$f_{it} \text{ scalars (risk factors)}$$

where the *matrix normal distribution* is

$$X \sim \text{MN}(M, U, V)$$

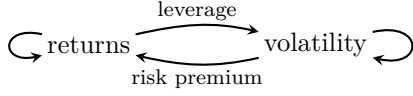
$$M: \text{ mean}$$

$$U: \text{ variance among rows}$$

$$V: \text{ variance among columns}$$

$$\text{vec } X \sim N(\text{vec } M, V \otimes U).$$

Use both stock returns and volatility, and separately model $A_{\text{ret,ret}}, A_{\text{ret,vol}}, A_{\text{vol,ret}}, A_{\text{vol,vol}}$.



Application: Bayesian inference, 500 European firms, pairwise Granger causality

Deep portfolio optimization via distributional prediction and residual factors
K. Imajo et al. (2020)

Do not forecast returns, but idiosyncratic returns – residuals, ε , of a factor model.

Forecast the return distribution using quantile regression: a neural network forecasting the quantiles of $\varepsilon_{i,t} | \varepsilon_{i,<t}$ (assume $\Sigma = \text{Var } \varepsilon$ is diagonal).

Account for inductive biases:

- Amplitude invariance (volatility clustering): $(x_t)_t$ and $(\lambda x_t)_t$ are equally likely, using homogeneous function $x \mapsto \|x\| \phi(x/\|x\|)$;
- Time-scale invariance: $(x_t)_t$ and $(x_{\lambda t})_t$ are equally likely, using shared weights at different scales.

Maximize the Sharpe ratio of the optimal portfolio $w = \lambda^{-1} \Sigma^{-1} \mu$.

Algorithms for learning graphs in financial markets
J.V.M. Cardoso et al. (2020)

The graphical lasso models financial returns as

$$\underset{\Sigma^{-1} \succcurlyeq 0}{\text{Minimize}} \text{tr}(\Sigma^{-1} S) - \log \det^*(\Sigma^{-1}) + \alpha \|\Sigma^{-1}\|_1.$$

Add a few constraints:

- The precision matrix Σ^{-1} is a graph Laplacian, $\Sigma^{-1} = \mathcal{L}W = \text{diag}(W\mathbf{1}) - W$;
- The graph has k connected components, i.e., $\text{rank } \Sigma^{-1} = p - k$;
- To avoid isolated nodes, each node has degree d , $\text{diag } W = d\mathbf{1}$.

The problem can be solved by alternatingly estimating Σ^{-1} and W (with ADMM).

$$\begin{array}{ll} \text{Find} & W \geq 0, \Theta \geq 0 \\ \text{To minimize} & \text{trace}(S\mathcal{L}W) - \log \det^* \Theta \\ \text{Such that} & \Theta = \mathcal{L}W \\ & \text{rank } \Theta = p - k \\ & \text{rank } \mathcal{L}W = p - k \\ & \text{diag } W = d\mathbf{1} \end{array}$$

The rank constraint can be enforced with $\sum_{i=1}^k \lambda_i = 0$, using

$$\sum_{i=1}^k \lambda_i(A) = \underset{\substack{V \in \mathbf{R}^{p \times k} \\ V'V = I}}{\text{Min}} \text{tr}(V'AV)$$

For robustness to outliers, replace the Gaussian distribution with a Student (the problem is no longer convex, but still amenable to MM).

ADMM solves

$$\begin{array}{ll} \text{Find} & x, z \\ \text{To minimize} & f(x) + g(z) \\ \text{Such that} & Ax + Bz = c \end{array}$$

by iterating

$$\begin{array}{l} x \leftarrow \text{Argmin } L(x, z, y) \\ z \leftarrow \text{Argmin } L(x, z, y) \\ y \leftarrow y + \lambda(Ax + Bz - c) \end{array}$$

where $L(x, z, y)$ is

$$f(x) + g(z) + y'(Ax + Bz - c) + \frac{1}{2} \|Ax + Bz - c\|^2.$$

R implementation in `fingraph`.

Learning high-dimensional Gaussian graphical models under total positivity without adjustment of tuning parameters
Y. Wang et al. (2020)

A Gaussian distribution is multivariate totally positive of order 2 (MTP₂, attractive Gaussian random field) if all partial correlations are positive.

A density f on \mathbf{R}^p is MTP₂ if

$$\forall x, y \in \mathbf{R}^p \quad f(x)f(y) \leq f(x \wedge y)f(x \vee y)$$

(where \wedge and \vee denote the elementwise min and max).

The CMIT algorithm estimates a sparse graphical model, starting with a complete graph, and removing edge $i-j$ if $|\rho_{ij|S}| \leq \lambda$ for some $S \subset [1, p]$, $|S| \leq \eta$ (max degree).

It can be modified for MTP₂ models: remove edge $i-j$ if $\hat{\rho}_{ij|S \cup \{k\}} < 0$ for some $S \subset \mathcal{N}(i) \setminus \{j\}$, $|S| \leq \ell$, and some $k \notin S \cup \{i, j\}$; progressively increase ℓ ; only estimate $\hat{\rho}$ on a minibatch.




cgam: an R package for the constrained generalized additive model
C. Liao and M.C. Meyer

Monotonicity and convexity constraints for GAMs; allows qualitative (ordered) variables.

Disparity of clustering coefficients in the Holme-Kim network model
R.I. Oliveira (2016)

The **Holme-Kim** random graph model extends the Barabasi-Albert one with a triad formation step (with some fixed probability) after each preferential attachment.

The *local clustering coefficient* of a node is the proportion of neighbours that are connected. The local clustering coefficient of a graph is the average of the local clustering coefficients of its nodes.

The *global clustering coefficient* of a graph is the proportion of paths of length 2  that can be completed in a triangle . Those coefficients are different: .

The local clustering coefficient of the Holme-Kim random graph remains bounded away from zero, but the global one slowly ($1/\log n$) tends to zero.

The atlas for the aspiring network scientist
M. Coscia (2020)

760-page illustrated book

fastnet: an R package for fast simulation and analysis of large scale social networks
X. Dong et al. (2020)

The H-index of a network and its relation to degree and coreness
L. Lü et al. (2015)

The core decomposition of a graph is obtained by removing all nodes of degree 1 (repeat until there are none left – new ones may be created), then all nodes of degree at most 2, and so on – the *coreness* of a node is the step at which it was removed. The H-index of a node is the largest h such that there are at least h neighbours with degree at least h . Iterating the H-index operator, starting with the degree, converges to the coreness; the updates can be asynchronous.

Characterization of complex networks: a survey of measurements
L.F. Costa et al. (2006)

List of graph metrics

Graph evolution: densification and shrinking diameters
J. Leskovec et al. (2007)

Real-world graphs grow with time and exhibit:

- densification: $(\#edges) \propto (\#nodes)^a$, $a > 1$;
- shrinking: the effective diameter (90% quantile of the distance distribution) decreases.

The *community-guided attachment model* builds a graph on the leaves of a complete b -ary tree of depth t (representing nested communities), by adding an edge i – j with probability proportional to $c^{d(i,j)}$, where the distance is measured on the tree; for a dynamic model, one can use all the nodes instead of just the leaves. It exhibits densification, but not shrinking.

The *forest fire model* adds nodes one by one, linking them to a random existing node, then to a random subset of its neighbours, then to a random subset of these neighbours' neighbours, and so on (citation networks arise like that). For a narrow range of parameters, it exhibits a heavy tail degree distribution, densification, shrinking, communities.

ANF: a fast and scalable tool for data mining in massive graphs
C.R. Palmer et al. (2002)

To approximately compute the size of neighbourhoods,

$$\mathcal{N}(x, h) = \{y \in V : d(x, y) \leq h\}$$

iterate (for all x , and increasing h)

$$\mathcal{N}(x, h) = \bigcup_{x-y} \mathcal{N}(y, h-1)$$

using HyperLogLog structure (to count distinct elements). Use approximate neighbours to speed computations further.

New methods for generating synthetic equivalents of real social networks
D. Xu et al. (2018)

Generate a random network with overlapping communities by assigning each node to a random number of random communities and connecting all the nodes in each community. Prune and/or rewire if desired.

Polychrome: creating and assessing qualitative palettes with many colours
K.R. Coombes et al. (2019)

If you need large palettes (up to 36 colours).

vsgoftest: and R package for goodness of fit testing based on KL divergence
J. Lequesne and P. Regnault (2020)

The Vasicek normality test relies on the Gaussian distribution maximizing entropy (for a fixed variance). The Vasicek-Song gof test generalizes it to arbitrary (continuous) distributions, by using the KL divergence between the sample and the null distributions.

Flexible regression models for count data based on renewal processes: the Countr package
T. Kharat et al. (2019)

Count data is often modeled as a Poisson distribution (or, sometimes, Geometric or Binomial), but this imposes $EY = \text{Var} Y$, has no memory, and constant hazard function. Instead, consider renewal processes with other inter-event distributions (Weibull, Gamma, Burr).

```
r <- renewalCount(y~1, d, "weibull")
```

Evaluating probabilistic forecasts with scoring rules
A. Jordan et al. (2019)

Scoring rules compare probabilistic forecasts: let F be the forecast (a probability distribution, parametric or given by samples), y the outcome, and $S(F, y)$ the score (lower is better).

A proper scoring rule satisfies

$$\forall F, G \quad \mathbb{E}_{Y \sim G} S(G, Y) \leq \mathbb{E}_{Y \sim G} S(F, Y),$$

with equality iff $F = G$. Examples include the logarithmic score and the *continuous ranked probability score* (CRPS)

$$\begin{aligned}\text{LogS}(F, y) &= -\log f(y) \\ \text{CRPS}(F, y) &= \int [F(z) - \mathbf{1}_{y \leq z}]^2 dz \\ &= \mathbb{E}_{X \sim F} |X - y| - \mathbb{E}_{X_1, X_2 \sim F} |X_1 - X_2|.\end{aligned}$$

If the forecast is provided as a sample, the CRPS can be efficiently computed from the order statistics.

Bayesian and non-Bayesian cause-specific competing risk analysis for parametric and non-parametric survival functions: the R package CFC
A.S. Mahani and M.T.A. Sharabiani

Cause-specific competing risk survival analysis is survival analysis with multiple, mutually-exclusive endpoints: $\hat{F}_k(t) = [1 - \sum F_\ell(t)] \lambda_k(t)$. By letting $E(t) = 1 - \sum F_k(t)$, we get $\dot{E}/E = -\sum \lambda_k$ and

$$\begin{aligned}E &= \prod S_k \\ S_k(t) &= \exp - \int_0^t \lambda_k(s) ds \\ F_k(t) &= \int_0^t \prod S_\ell(s) \lambda_k(s) ds.\end{aligned}$$

Randomized matrix decompositions using R
N.B. Erichson et al. (2019)

The `rsvd` package provides randomized (for big data) SVD, PCA, robust PCA, CUR, interpolative decomposition. Also check `irlba`, `RSpectra`, `svd::propack.svd`, `svd`.

ManifoldOptim: an R interface to the ROPTLIB library for Riemannian manifold optimization
S.R. Martin et al. (2020)

The *minimum average deviance estimation* (MADE) method for sufficient dimension reduction (SDR) looks for B such that the local regression (with weights from a kernel in the original space, and bandwidth from cross-validation) $y \sim B'X$ gives the best fit: it is an optimization on the Grassmanian.

In a linear model $Y = \mu + \beta X + \varepsilon$, $X \in \mathbf{R}^p$, $Y \in \mathbf{R}^r$, $\varepsilon \sim N(0, \Sigma)$, not all of Y contains relevant information: the Σ -envelope of $\text{Span } \beta$ is the smallest subspace $S \supset \text{Span } \beta$ such that $\Sigma = \Sigma_S + \Sigma_{S^\perp}$. Finding such a subspace can be formulated as an optimization problem on the Grassmanian.

Also check `Pymanopt` (Python), `GrassmanOptim` (R), `ROPTLIB` (C++, in Julia, or R: `ManifoldOptim`, `ldr` (likelihood based dimension reduction)).

An introduction to the augmented inverse propensity weighted estimator
A.N. Glynn and K.M. Quinn (2009)

The average treatment effect (ATE) is $\mathbb{E}[Y(1) - Y(0)]$. There are many more acronyms: ATT, ATC (average treatment effect on treated/control), SATC, PATE (sample/population ATE), CATE (conditional ATE – for a subgroup).

- *Propensity matching* is not data-efficient;
- The *regression estimator* fits a model $Y \sim T + Z$,

$$\widehat{\text{ATE}} = \text{Mean}_i [\hat{Y}(T = 1, Z = z_i) - \hat{Y}(T = 0, Z = z_i)]$$

- The *inverse propensity weighting* (IPW) estimator models the probability of treatment $\pi(z)$,

$$\widehat{\text{ATE}} = \text{Mean}_i \left[\frac{T_i}{\pi(Z_i)} Y_i - \frac{1 - T_i}{1 - \pi(Z_i)} Y_i \right]$$

It has poor small-sample properties (π can be close to zero) and can even be outside the possible range for $Y(1) - Y(0)$.

- IPW* normalizes the weights to ensure they sum to 1.

$$\begin{aligned}\widehat{\text{ATE}} &= \left(\sum \frac{T_i}{\pi(Z_i)} \right)^{-1} \sum \frac{T_i}{\pi(Z_i)} Y_i \\ &\quad - \left(\sum \frac{1 - T_i}{1 - \pi(Z_i)} \right)^{-1} \sum \frac{1 - T_i}{1 - \pi(Z_i)} Y_i\end{aligned}$$

- The covariate Z contains information, not only about T , but also about the outcome Y . The *augmented IPW* (AIPW) adjusts the ATE accordingly. It is *doubly robust*: it is consistent if either π or \hat{Y} is well specified. R implementation in `CausalGAM`.

Debiased inverse propensity score weighting for estimation of average treatment effects with high dimensional confounders
Y. Wang and R.D. Shah (2020)

The oracle estimator

$$\text{ATE} = \text{Mean}_i \frac{T_i}{\pi(Z_i)} Y_i - \text{Mean}_i \frac{1 - T_i}{1 - \pi(Z_i)} Y_i$$

is unbiased, but there is a bias of the propensity score is estimated from data.

If $\mu \perp\!\!\!\perp T|Z$,

$$\text{ATE} = \text{Mean}_i \frac{T_i}{\pi(Z_i)} (Y_i \mu_i) - \text{Mean}_i \frac{1 - T_i}{1 - \pi(Z_i)} (Y_i - \mu_i)$$

has the same properties, and we can choose μ to reduce the bias, e.g., with

$$\begin{aligned}\text{Find } & \mu \\ \text{To minimize } & \sum (\hat{Y}(Z = Z_i) - \mu_i)^2 \\ \text{Such that } & \left\| \frac{1}{n} Z'(\tilde{Y} - \tilde{f}(Z)) - \frac{1}{n} Z'(\mu - \tilde{f}(Z)) \right\|_\infty \leq \eta\end{aligned}$$

where

$$\tilde{Y} = \frac{T(1 - \hat{\pi})}{\hat{\pi}} Y + \frac{(1 - T)\hat{\pi}}{1 - \hat{\pi}} Y$$

(to ensure $\mu \perp T|Z$, split the data in 3: to estimate $\hat{\pi}$ and $\tilde{f} = \hat{Y}$, for the constraint, and for the rest).

Other methods include: IPW, AIPW, ARN (approximate residual rebalancing), TMLE (targeted MLE).

Zigzag expanded navigation plots in R: the R package zenplots

Zigzag plots are pairs plots where adjacent axes share the same variable.

Latent space approaches to social network analysis **P.D. Hoff et al. (2002)**

In a latent space network model, nodes have both features x_i and latent coordinates z_i , and the edge probabilities are

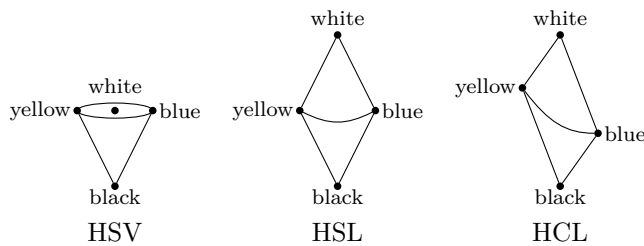
$$z_i \sim N(\mu, \Sigma)$$

$$\text{logit } y_{ij} = \beta x_i + \beta x_j + d(z_i, z_j)$$

GNAR: Generalized network autoregressive process in R **M. Knight et al. (2020)**

In GNARs, X_{it} is a linear combination of its past values (AR) and those of its neighbours (GCN), with weights depending on the lag and the distance (number of hops).

colorspace: a toolbox for manipulating and assessing colours and palettes **A. Zeilis et al. (2020)**



Tensor learning for regression **W. Guo and I. Patras (2012)**

A *tensor regression* is a linear regression with tensors, e.g., $Y_i = X_{ijk\ell} W_{jkl} + b$ where the weight tensor W has low weight (e.g., a CP decomposition – use alternating optimization)

TensorLy: tensor learning for Python **J. Kossaifi et al. (2019)**

ETAS: an R package for fitting the space time ETAS model of earthquake data

The ETAS (epidemic-type aftershock sequence) marked point process is a Hawkes-like model for earthquakes (time, position, magnitude).

Just interpolate: kernel ridge regression can generalize **T. Liang and A. Rakhlin (2018)**

In ridge regression

$$\underset{\beta}{\text{Minimize}} \|y - X\beta\|^2 + \lambda \|\beta\|^2$$

we can let $\beta \rightarrow 0$. If the model is overparametrized, the OLS solution is not unique and interpolates the data; the problem becomes

$$\begin{array}{ll} \text{Find} & \beta \\ \text{To minimize} & \|\beta\|^2 \\ \text{Such that} & y = X\beta. \end{array}$$

Surprises in high-dimensional ridgeless least squares interpolation **T. Hastie et al. (2020)**

Interpolation does not contradict generalization.

A continuous time view of early stopping for least squares **A. ALi et al. (2019)**

Early stopped gradient approximates ridge regression: with the gradient flow and $t = 1/\lambda$, the error is at most 1.7 times that of the ridge.

$$\begin{array}{ll} \beta \leftarrow \beta + \varepsilon \frac{X'}{n} (y - X\beta) & \text{Gradient descent} \\ \dot{\beta} = \frac{X'}{n} (y - X\beta) & \text{Gradient flow} \\ \beta(\lambda) = (X'X - n\lambda I)^{-1} X'y & \text{Ridge} \\ \beta(t) = (X'X)^\dagger (I - \exp -tX'X/n) X'y & \end{array}$$

Computing the Oja median in R: the package OjaNP **D. Fischer et al. (2020)**

There are several multivariate generalizations of the median:

- The *marginal median* is the coordinate-wise median; it is not affine equivariant;
- The *Tuckey median* is the intersection of the minimal halfplanes containing at least half the data;
- The *spatial median* μ minimizes $\sum_i \|x_i - \mu\|_2$; it is not affine-equivariant;
- The *Oja median* is

$$\underset{\mu}{\text{Argmin}} \sum_{i_1 < \dots < i_k} V(x_{i_1}, \dots, x_{i_k}, \mu)$$

where $V(x_1, \dots, x_{k+1})$ is the volume of the simplex (x_1, \dots, x_{k+1}) in \mathbf{R}^k .

They define *signs*

$$\begin{aligned}\text{msgn } x &= [\text{sign } x_1, \dots, \text{sign } x_k] \\ \text{msgn}_X x &= \text{msgn}(x - \text{mmed } X) \\ \text{ssgn } x &= \frac{x}{\|x\|} \text{ if } x \neq 0 \\ \text{ssgn}_X x &= \text{ssgn}(x - \text{smed } X) \\ \text{osgn}_X x &= \sum_{i_1 < \dots < i_{k-1}} \nabla_x \det[x_{i_1} - m, \dots, x_{i_{k-1}} - m, x - m]\end{aligned}$$

and ranks $\frac{1}{n} \sum \text{sign}(x_i - x)$, and rank (or sign) correlation matrices.

Covatest: an R package for selecting a class of space-time covariance functions
C. Cappello et al. (2020)

Classes of space-time covariance functions include:

- Sum-product: the increments of C , in the space and time directions, are constant

$$C(\mathbf{h}, u) = k_1 C_0(\mathbf{h}) C_1(u) + k_2 C_2(\mathbf{h}) + k_3 C_3(u);$$

- Integrated product: the increments of $1/C$ are constant;
- Gneiting: the increments of $\log C$ are constant.

BayesNetBP: an R package for probabilistic reasoning in Bayesian networks
H. Yu et al. (2020)

Exact inference in directed probabilistic graphical models (PGM), assuming the structure known (use `bnlearn`, `deal`, `catnet`, `pcalg` if needed) – `gRain` is limited to discrete Bayesian networks.

mlogit: random utility models in R
Y. Croissant (2020)

R package for generalizations of the multinomial logit/probit model (modeling discrete choices driven by n unobserved utility function): heteroskedasticity, correlated errors, individual heterogeneity (mixed models).

Pseudo-counts: how to compute them efficiently in R
M. Happ et al. (2020)

Comparing two or more samples with rank-based tests can lead to paradoxical results (e.g., $\mu_1 < \mu_2 < \mu_3 < \mu_1$): replace the ranks $\frac{1}{a} \sum \hat{F}(x)$ with the pseudo-ranks $\hat{F}(x)$ where

$$\begin{aligned}Y &\sim \text{Categorical}([1, a], p) \\ X &\sim F_Y \\ F &= \sum p_i F_i\end{aligned}$$

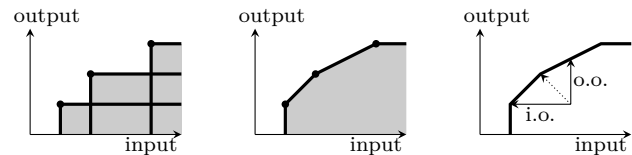
A data envelopment analysis toolbox for Matlab
I.C. Álvarez et al. (2020)

Data envelopment analysis (DEA) is a way of measuring the “distance” (of a firm, turning some input into some output) to the efficient frontier.

$$\begin{aligned}n: \# \text{ firms} & & X \in \mathbf{R}^{m \times n} \text{ inputs} \\ m: \# \text{ inputs} & & Y \in \mathbf{R}^{s \times n} \text{ outputs} \\ s: \# \text{ outputs}\end{aligned}$$

Let $P = \{(x, y) : \exists \lambda \in \mathbf{R}_+^n \ x \geq X\lambda, \ y \leq Y\lambda, \ \lambda' \mathbf{1} = 1\}$ be the set of attainable solutions (remove $\lambda' = 1$ is everything is scalable).

The *radial input-oriented model* measures the efficiency of firm i as $\theta_i = \text{Min}\{\theta : (\theta x_i, y_i) \in P\}$. If $\theta_i < 1$, we can improve (x_i, y_i) (produce the same with less). The *input excess* and *output shortfall* are $\theta x - X\lambda$ and $Y\lambda - y$. Similarly, the radial output oriented model is $\phi_i = \text{Max}\{\phi : (x_i, \phi y_i) \in P\}$. There are also directional variants.



The real numbers – a survey of construction
I. Weiss (2015)

Cauchy sequences and Dedekind cuts are not the only constructions of \mathbf{R} :

- Decimal expansions (but arithmetic is tricky); $\sum a_n/2^n$ with no restriction on a_n (up to equivalence);
- Nested families of intervals; Cauchy filters;
- $\sum_{s \in S} s$, where S is a multiset of elements of the form $1/n$; $\sum_{n \in A} 1/n$, $\sum_{n \in A} a_n$ for a fixed sequence a such that $a_n \rightarrow 0$, $\sum a_n = \infty$;
- Continued fractions; product $\prod (1 + 1/a_n)$; $a_0 + 1/a_1 + 1/a_2 + \dots$; $a_0 + 1/a_1 + 1/a_1 a_2 + 1/a_1 a_2 a_3 + \dots$;
- Approximate endomorphisms, *i.e.* $f : \mathbf{Z} \rightarrow \mathbf{Z}$ such that $\{f(x+y) - f(x) - f(y)\}$ be finite, up to equivalence $f \sim g$ if $\{f(x) - g(x)\}$ is finite;
- The universal property of the completion of \mathbf{Q} .

Fisher information distance: a geometric reading
S.I.R. Costa et al. (2014)

Formulas for computations on the Riemannian manifold of Gaussian distributions $N(\mu, \sigma^2)$.

Self-normalizing flows
T.A. Keller et al. (2020)

Normalizing flows require the Jacobian determinant, which is expensive to compute, unless it is triangular or easy to approximate – instead, have the network learn

both the flow at each layer and its inverse [cf. synthetic gradients].

***Explaining by removing:
a unified framework for model explanation***
I.C. Covert et al. (2020)

Most model explanation methods (SHAP, LIME, etc.) can be unified and presented as a 3-step process:

- Removal: remove part of the image (one or several circular regions, or regions from a segmentation algorithm) and replace it (with noise, a constant value, a generative model, blur, etc.);
- Behaviour: compute the effect of the removal (on the output, the loss, the average loss);
- Aggregation: aggregate those effects (smallest destroying subset, smallest sufficient subset, mean after inclusion/removal, Shapley value, linear model, etc.).

***Reservoir computing meets
recurrent kernels and structured transforms***
J. Dong et al. (2020)

A function $f : \mathbf{R} \rightarrow \mathbf{R}^k$ and random points w_i in \mathbf{R}^p (sampled from some distribution) define random features

$$\phi(u) = \frac{1}{\sqrt{N}} [f(\langle w_1, u \rangle), \dots, f(\langle w_N, u \rangle)]$$

and a kernel $k(u, v) = \langle \phi(u), \phi(v) \rangle$.

***C-space tunnel discovery
for puzzle path planning***
X. Zhang et al. (2020)

Rigid body disentanglement puzzles rely on narrow tunnels in configuration space. Find those tunnels by aligning the features (gap-gap or gap-notch) of the pieces:

- Euclidean-to-geodesic ratio pairs of points on a surface where the EGR is a *local* minimum are around gaps or notches;

$$\text{Minimize}_{u,v} \frac{d_E(u,v)}{d_G(u,v)} + \alpha d_E(u,v)$$

use the *heat kernel method* to compute the geodesics;

- Medial axis notch detection:
 - Compute the medial axis (a surface);
 - Use the *mean curvature flow* to reduce it to a curve, the skeleton;
 - For each point on the skeleton, find the closest point on the surface, and compute the distance to the opposite point;
 - Notches are local minima for this distance;
- Neural network, trained on data from simple puzzles (solved by a traditional planner), taking an image of the puzzle as input (5 channels: luminance, depth, normal) and predicting where the tunnels are.

This does not work on puzzles not relying on gap-gap or gap-notch alignment (e.g., Hanamaya Elk).

***Fast and flexible temporal point processes
with triangular maps***
O. Schur et al. (2020)

A *triangular map* is a map $f : \mathbf{R}^N \rightarrow \mathbf{R}^N$ such that $f_i(x_1, \dots, x_n)$ only depends on (x_1, \dots, x_i) and f_i is increasing in x_i . They are used in normalizing flows: if $x \sim p$, then $F(x) \sim \tilde{p}$, with

$$p(x) = |\det J_F(x)| \tilde{p}(F(x)) = \left(\prod \frac{\partial f_i}{\partial x_i} \right) \tilde{p}(F(x)).$$

This formula is strangely similar to the likelihood of a temporal point process (TPP)

$$\begin{aligned} p(t) &= \prod \lambda(t_i) \exp - \int_0^T \lambda(u) du \\ &= \left(\prod \frac{d\Lambda}{dt}(t_i) \right) \exp -\Lambda(T) \end{aligned}$$

where $\Lambda(T) = \int_0^T \lambda(u) du$.

To sample, draw z from a homogeneous Poisson process on $[0, \Lambda(T)]$ and compute $t = F^{-1}(z)$.

Examples include:

- Inhomogeneous Poisson processes: Λ is deterministic and $F = \Lambda$ (elementwise);
- Renewal processes: $\Lambda(t) = \Phi(t - t_1) + \sum_{j \leq t} \Phi(t_j - t_i)$, where t_i is the last event before t , and $\Lambda = C \circ \Phi \circ D$, where

$$D = \begin{pmatrix} 1 & & 0 \\ -1 & & \\ & \ddots & \\ 0 & -1 & 1 \end{pmatrix}$$

computes pairwise differences, $C = D^{-1}$ (cumulated sum) and Φ acts elementwise.

- Modulated renewal process:

$$\Lambda(t) = \Phi(\Lambda(r) - \Lambda(t_i)) + \sum_{j \leq t} \Phi(\Lambda(t_j) - \Lambda(t_{j-1}))$$

and $F = C\Phi D\Lambda$.

Use rational quadratic splines for Φ and Λ .

Generalize the model to $F = C\Phi_2 B_L \dots B_1 \Phi_1 D\Lambda$ where the B_ℓ are lower-triangular block-diagonal matrices.

Eigengame: PCA as a Nash equilibrium
I. Gemp et al. (2020)

To find the first eigenvector of a symmetric matrix $M = X'X$, one can solve the optimization problem

$$\begin{aligned} &\text{Find} && v \\ &\text{To maximize} && \frac{1}{2} v' M v \\ &\text{Such that} && \|v\| = 1 \end{aligned}$$

with Oja's rule (Krasulina's algorithm):

Mv	gradient
$(I + vv')Mv$	projection of Mv on TS^{n-1}
$v \leftarrow v + \eta(I + vv')Mv$	gradient step
$v \leftarrow v / \ v\ $	Projection on S^{n-1}

This can be generalized to the top k eigenvectors: replace \mathbf{S}^{n-1} with the Stiefel manifold and use Gram-Schmidt orthonormalization (matrix Krasulina).

We can recover the span of the top k eigenvectors by solving

$$\begin{array}{ll} \text{Find} & V \in \mathbf{R}^{n \times k} \quad (k < n) \\ \text{To maximize} & \text{tr}(V' M V) \\ \text{Such that} & V' V = I. \end{array}$$

For $k = n$, we want to maximize the trace, minimize the off-diagonal elements; to recover the order of the eigenvectors, each is only penalized by the previous ones.

$$\begin{aligned} v_1 &\leftarrow \underset{v_1' v_1 = 1}{\text{Argmax}} \langle v_1, M v_1 \rangle \\ v_2 &\leftarrow \underset{v_2' v_2 = 1}{\text{Argmax}} \langle v_2, M v_2 \rangle - \frac{\langle v_2, M v_1 \rangle^2}{\langle v_1, M v_1 \rangle} \\ v_i &\leftarrow \underset{v_i' v_i = 1}{\text{Argmax}} \langle v_i, M v_i \rangle - \sum_{j < i} \frac{\langle v_i, M v_j \rangle^2}{\langle v_j, M v_j \rangle} \end{aligned}$$

(the constraints $v_i' v_j = 0$ are redundant: $\langle v_i, M v_j \rangle = \lambda \langle v_i, v_j \rangle$ at the optimum.) The i th objective can also be written

$$\begin{aligned} u_i(v_i) &= \langle v_i, M v_i \rangle - \sum_{j < i} \frac{\langle v_i, M v_j \rangle^2}{\langle v_j, M v_j \rangle} \\ &= \|X v_i\|^2 - \sum_{j < i} \frac{\langle X v_i, X v_j \rangle^2}{\langle X v_j, X v_j \rangle}. \end{aligned}$$

PCA is the unique strict Nash equilibrium of this k -player differentiable game. Solve with Riemannian gradient ascent on the sphere, sequentially or (asynchronously) in parallel.

Learning undirected graphs in financial markets

J.V.M. Cardodo and D.P. Palomar (2020)

Do not learn a graph but its Laplacian $L = D - W$

$$\begin{array}{ll} \text{Find} & L \\ \text{To minimize} & \text{tr}(L S) - \log \text{gdet } L + h_\alpha(L) \\ \text{Such that} & L \mathbf{1} = 0 \\ & \forall i \neq j \quad L_{ij} = L_{ji} \leq 0 \end{array}$$

where S is the (detoned) correlation and gdet the pseudo-determinant (the product of the nonzero (positive) eigenvalues).

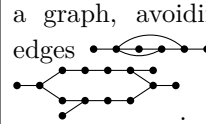
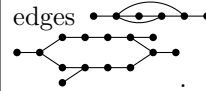
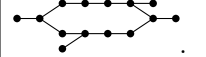
To constrain the number of connected components $k = \dim \ker L$, note that

$$\begin{aligned} \dim \ker L \geq k &\iff \sum_{i=1}^k \lambda_i(L) = 0 \\ \sum_{i=1}^k \lambda_i(L) &= \underset{\substack{V \in \mathbf{R}^{p \times k} \\ V' V = I}}{\text{Min}} \text{tr}(V' L V). \end{aligned}$$

The resulting problem can be solved in an alternating fashion.

$$\begin{array}{ll} \text{Find} & L \succcurlyeq 0 \\ & V \in \mathbf{R}^{p \times k} \\ \text{To minimize} & \text{tr}(L S) - \log \text{gdet } L + \eta \text{tr}(V' L V) \\ \text{Such that} & L \mathbf{1} = 0 \\ & \forall i \neq j \quad L_{ij} = L_{ji} \leq 0 \\ & \text{diag } L = \mathbf{1} \\ & V' V = I \end{array}$$

A step towards neural genome assembly **L. Vrček et al. (2020)**

De novo genome assembly looks for a path in a graph, avoiding short tips , transitive edges  and choosing a branch in bubbles .

Use a GNN for each of those three problems.

Sanity checks for saliency maps **J. Adebayo (2018)**

It is difficult to assess the quality of model explanations. Saliency maps (gradient, gradient \odot input, integrated gradient, guided backprop, guided GradGAM, smoothGrad) are misleading: they often give results similar to a simple (task-agnostic) edge detector – compare them with an untrained model, or a model with shuffled labels.

Coupling-based invertible neural networks are universal diffeomorphism approximators **T. Teshima et al. (2020)**

A set \mathcal{M} of functions $\mathbf{R}^m \rightarrow \mathbf{R}^n$ is a *distributionally universal approximator* if, for all absolutely continuous measure μ on \mathbf{R}^m , for all measure ν on \mathbf{R}^n , there exists a sequence $(g_i)_i$ in \mathcal{M} such that $g_{i*} \mu \xrightarrow{i \rightarrow \infty} \nu$.

Affine coupling flows are transformations of the form $\psi(x, y) = (x, e^{s(x)} \odot y + t(x))$ (the exponential is elementwise).

Combining affine flows (with y 1-dimensional) and invertible affine transformations gives an L^p universal approximator, and therefore a distributionally universal approximator.

Ignorance is bliss: adversarial robustness by design through analog computing and synaptic asymmetry **A. Cappelli et al. (2020)**

To fight adversarial attacks, insert a non-differentiable layer (an optical processing unit (OPU), $m = |R x|^2$, R random, input and output quantized to 8 bits) and use direct feedback alignment (DFA) instead of backpropagation to differentiate through it.

Hard negative mixing for contrastive learning
Y. Kalantidis et al. (2020)

Contrastive learning (learning an embedding in which different augmentations of the same image are closer than those of another image) benefits from better positive pairs (more data augmentation) or better negative pairs (SinCLR, MoCo): MoChi creates synthetic negatives by averaging two negative samples, or a negative and a positive sample.

SuperLoss:

a generic loss for robust curriculum learning

Early in training, downweight hard samples, identified by a large loss, (or a large loss with high confidence) by changing the loss function.

Deep transformation invariant clustering
T. Monnier et al.

Image clustering often relies on latent representations. Instead, work in pixel space, and jointly learn prototypes (for each cluster) and transformations (to align each image to its prototype – or the inverse). It is an image (parametric) analogue of dynamic time warp (DTW).

Clustering	$\underset{c_1 \dots c_K}{\text{Argmin}} \sum_i \ell(x_i, \{c_1, \dots, c_K\})$
TI clustering	$\underset{c_1 \dots c_K}{\text{Argmin}} \sum_i \underset{\beta_1 \dots \beta_K}{\text{Min}} \ell(x_i, \{T_{\beta_1} c_1, \dots, T_{\beta_K} c_K\})$
DTI clustering	$\underset{\substack{c_1 \dots c_K \\ f_1 \dots f_K}}{\text{Argmin}} \sum_i \ell(x_i, \{T_{f_1(x_i)} c_1, \dots, T_{f_K(x_i)} c_K\})$

The transformation parameters are computed by a neural network f . Each transformation is a sequence of parametric transformations $T_{\beta_1} \circ \dots \circ T_{\beta_M}$, including:

- Affine colour transformations;
- Spatial transformers networks;
- Morphological transformations, with the max and min of dilation and erosion replaced by softmax ($\alpha > 0$) and softmin ($\alpha < 0$).

$$T(x)_{uv} = \frac{\sum_{u'v'} x_{u+u',v+v'} \cdot a_{u+u',v+v'} \cdot \exp \alpha x_{u+u',v+v'}}{\sum_{u'v'} a_{u+u',v+v'} \cdot \exp \alpha x_{u+u',v+v'}}$$

The loss function can be from k -means of a Gaussian mixture (GMM). During training:

- Reassign tiny clusters to a noisy copy of the largest cluster;
- Progressively add more complex transformations (curriculum learning).

Image analysis using mathematical morphology
R.M. Haralich et al. (1987)

Spatial transformer network
M. Jaderberg et al. (2016)

Transform an image with a parametric transformation T_θ (affine, projective, peicewise affine, thin plate spline) with the parameters computed from the image itself:

$$x \mapsto T_{\theta(x)}(x).$$

Learning energy-based models by diffusion recovery likelihood
(ICLR 2021)

An energy-based model (EBM) $p_\theta(x) = f_\theta(x)/Z_\theta$ can describe a distribution on images x (use a CNN for f_θ): training is done with the gradient

$$\underset{x \sim \text{data}}{\text{E}} \frac{\partial f_\theta(x)}{\partial \theta} - \underset{x \sim p_\theta}{\text{E}} \frac{\partial f_\theta(x)}{\partial \theta}$$

and sampling with Langevin Monte Carlo

$$x_{t+1} \leftarrow x_t + \frac{\delta^2}{2} \nabla_x f_\theta(x_t) + \delta \varepsilon_t$$

but convergence is slow.

Recovery likelihood, from a noisy image $\tilde{x} = x + \sigma \varepsilon$ (or $\tilde{x} = ax + \sigma \varepsilon$, a, σ constants) is easier to sample from (less multimodal)

$$p(x|\tilde{x}) = \frac{\exp \left[f_\theta(x) - \frac{\|\tilde{x} - x\|^2}{2\sigma^2} \right]}{\tilde{Z}_\theta(\tilde{x})}$$

Use a sequence of EBMs, trained on increasingly noise versions of the dataset.

Score-based generative modeling through stochastic differential equations
(ICLR 2021)

Score-based generative models consider a sequence of noise scales $\sigma_1 < \dots < \sigma_N$, going from the data distribution p_{data} to near-Gaussian noise $p_{\text{data}} * N(0, \sigma_N I)$, and learn a score function s_θ minimizing

$$\sum_i \sigma_i^2 \underset{s \sim \text{Data}}{\text{E}} \underset{\tilde{x} \sim p_{\sigma_i}(\tilde{x}|x)}{\text{E}} \|s_\theta(\tilde{x}, \sigma_i) - \nabla_{\tilde{x}} \log p_{\sigma_i}(\tilde{x}|x)\|^2$$

Stochastic differential equations (SDE) provide a continuous generalization, progressively adding noise to an image with a diffusion

$$dx = f(x, t)dt + g(t)dW$$

and removing it with the *reverse-time SDE*

$$dx = [f(x, t) - g^2(t) \nabla_x \log p_t(x)]dt + g(t)dW.$$

**Reinforcement learning upside down:
don't predict rewards, just map them to actions**
J. Schmidhuber (2019)

UDRL learns a mapping

$$(\text{state}, \text{horizon}, \text{reward}) \mapsto \text{action}$$

using training data from the past

$$(s_1, t_2 - t_1, r_{[t_1, t_2]}, a_{[t_1, t_2]})$$

and extrapolate (use it to find actions leading to a higher reward in a shorter time frame).

**Extrapolation towards imaginary 0-nearest
neighbour and its improved convergence rate**
A. Okuno and H. Shimodaira (2020)

The k -nearest neighbour classifier is a weighted average, often with positive constant weights – but real weights can remove the bias and improve convergence.

Instead, the multiscale k -NN uses adaptive weights, by extrapolating the values as $k \rightarrow 0$, with a linear regression k -NN $\sim \text{distance}_k^2$.

**It is not what machines can learn,
it is what we cannot teach**
G. Yehuda et al. (2020)

Deep learning cannot solve NP-complete problems from data: that would require non-polynomial data generation.

**Movement pruning:
adaptive sparsity by fine-tuning**
V. Sanh et al. (2020)

For transfer learning, do not prune small weights, but weights that move towards zero when fine-tuning.



**On ranking via sorting
by estimated expected utility**
C. Calauzènes and N. Usunier (2020)

Learning-to-rank loss functions include

- The discounted cumulated gain

$$\text{DCG}_p = \sum_{i=1}^p \frac{\text{relevance}_i}{\log(1+i)}$$

- The expected reciprocal rank, the expectation of the inverse of the rank of the first relevant result, $E[1/\text{rank}]$, which encourages diversity.

A loss is compatible with expected utility iff the set of optimal ranking is connected (by transpositions of adjacent items).

**Tight nonparametric convergence rates
for stochastic gradient descent
under the noiseless linear model**
R. Berthier et al. (2020)

Kernel SGD performs interpolation as

$$f_k \leftarrow f_{k-1} - \gamma(f_{k-1}(x_k) - y_k)k(\cdot, x_k).$$

**Leverage the average:
an analysis of KL regularization in RL**
N. Vieillard et al. (2020)

Add the following penalties:

- Entropy, $\tau H(\pi)$, to penalize deterministic policies;
- Kullback-Leibler $-\lambda \text{KL}(\pi \parallel \pi_k)$, to remain close to the previous policy.

Munchausen reinforcement learning
N. Vieillard et al. (2020)

Augment the reward with the log-policy:

$$r(s, a) \rightsquigarrow r(s, a) + \alpha \log \hat{\pi}(s|a).$$

For the optimal policy:

$$\log \pi^*(a|s) = \begin{cases} 0 & \text{if } a \text{ is optimal} \\ -\infty & \text{otherwise.} \end{cases}$$

**Neumiss networks: differentiable programming
for supervised learning with missing values**
M. Le Morvan et al. (2020)

Under Gaussian assumptions, the linear model with missing values (MCAR, MAR – MNAR is more complex and requires more assumptions) is

$$f(x_{\text{obs}}) = \langle \beta_{\text{obs}}, x_{\text{obs}} \rangle + \langle \beta_{\text{mis}}, \mu_{\text{mis}} + \Sigma_{\text{mis, obs}} \Sigma_{\text{obs}}^{-1} (x_{\text{obs}} - \mu_{\text{obs}}) \rangle.$$

The model can be estimated with a neural network:

- β, μ, Σ are parameters to estimate;
- At each step, multiply elementwise the vectors with the missingness pattern m or $1 - m$, and matrices by mm' ;
- For Σ^{-1} , use a Neumann series, *i.e.*, $\Sigma^{-1} = \sum_{k \geq 0} (I - \Sigma)^k$, but learn the weights W (instead of using $W - I - \Sigma$).

$$\begin{aligned} S_0 &= W_0 \\ S_1 &= (W_1 \odot mm')S_0 + I \\ S_2 &= (W_2 \odot mm')S_1 + I \\ S_2 \odot mm' &\approx \Sigma_{\text{obs}}^{-1} \end{aligned}$$

**Statistical and topological properties
of sliced probability divergences**
K. Nadjahi et al. (2020)

The sliced Wasserstein distance (average of the Wasserstein distances after projection on random 1-dimensional subspaces) has good theoretical properties (similar to, and sometimes even better than the original Wasserstein distance).

The *generalized sliced Wasserstein distance* uses a non-linear projection, e.g., a neural network with a 1-dimensional output.

**The dynamics of learning
with feedback alignment**
M. Refinetti et al. (2020)

Backpropagation propagates the error ε of a neural net from the output to the input: each layer receives this error, transformed by a product of matrices $A_{\ell+1}A_{\ell+2} \cdots A_n\varepsilon$.

Instead, **direct feedback alignment** (DFA) uses fixed random matrices.

Evidence from shallow networks shows two phases:

- First, the weights change until the DFA direction becomes aligned with the actual gradient;
- Only then does learning actually start.

**Direct feedback alignment scales to modern
deep learning tasks and architectures**
J. Launay et al. (2020)

Extensive survey of DFA on complex architectures: they fail with CNNs, struggle with graph neural nets, but work surprisingly well with everything else (fully-connected networks, attention, etc.).

**Low-dimensional hyperbolic
knowledge graph embedding**
I. Chami et al. (2020)

A knowledge graph encodes different types of relations: hierarchical (e.g., “is a”), symmetric (e.g., “married to”), asymmetric, etc.

Embed both entities and relations in hyperbolic space, with a *learned* curvature c , different for each type of relation.

For the relations r , do not only learn an embedding, but also a rotation Rot_r , and a reflection Ref_r ; apply them to the head h of the triplet (h, r, t) ; combine them with an attention mechanism, and translate (Möbius addition) with the relation embedding

$$\text{Att}(\text{Rot}_r \cdot h, \text{Ref}_r \cdot h) \oplus^c r.$$

Data from WorldNet, Freebase, Yago3.

The (non-commutative, non-associative) Möbius addition is defined from the parallel transport:

$$x \oplus y = \exp_x P_{0 \rightarrow x} \log_0 y.$$

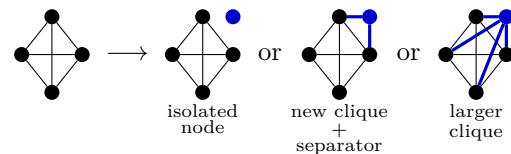
Learning clique forests
G.P. Massara and T. Aste (2019)

A *clique* is a maximal complete subgraph. A *separator* is an intersection of two cliques. A *clique graph* has cliques as nodes and separators as edges. It is a **clique forest** if it has no cycles (forest) and satisfies the “clique intersection property”: if C_1 and C_2 are cliques, then $C_1 \cap C_2$ is in all the cliques on the path from C_1 to C_2 .

A *perfect elimination order* is an ordering $\sigma = (v_1, \dots, v_n)$ of the vertices such that each $\text{Adj}(v_i) \cap G_{i:n}$ is a clique in $G_{i:n}$, where $\text{Adj}(v)$ is the set of vertices adjacent to v .

A graph is *chordal* iff it has a clique forest iff it has a perfect elimination order. The junction tree algorithm provides efficient exact inference on chordal undirected probabilistic graphical models (chordal Markov random fields).

The **maximally filtered clique forest** (MFCF) is an alternative to the graphical lasso (or other PGM structure inference algorithms: PC, etc.) which progressively builds a clique forest by adding vertices one by one with *clique expansion* to maximize the gain (increase in the “score”, e.g., the sum of the edge weights (correlations), or the log-likelihood).



Complex networks on hyperbolic surfaces
T. Aste et al. (2004)

Filter a graph by progressively adding edges as long as it remains embeddable in an orientable surface of genus g .

The genus of a graph (minimum genus of an orientable surface on which the graph can be embedded without edge intersections) is a measure of complexity; it is higher for small-world networks (which have “worm-holes”).

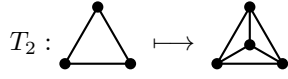
**A tool for filtering information
in complex systems**
M. Tuminello et al. (2005)

Idem.

**Network filtering for big data:
triangulated maximally filtered graph**
G.P. Massara et al. (2015)

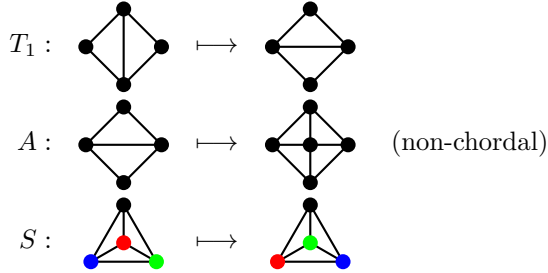
The planar maximally filtered graph (PMFG) can be constructed greedily, but with p planarity tests at $O(p^2)$, it is still $O(p^3)$.

The **deltahedron heuristic** starts with a tetrahedron $K_4 = \triangle$ and progressively adds vertices \triangleright .



Choose the triangle to maximize the sum of the edge weights; add the vertices in some predetermined order, e.g., using the sum or maximum of the incident edges. The result is a tree of 4-cliques separated by 3-cliques.

The *triangulated maximally filtered graph* (TMFG) chooses the optimal vertex and triangle at each step; it is still $O(p^2)$. It can be extended with more local moves



Topological regularization with information filtering networks T. Aste

The maximally filtered clique forest (MFCF), from the squared Kendall correlation, with different maximum clique sizes, can be used to infer the sparsity structure of the concentration matrix of a multivariate Gaussian or Student distribution, as an approximate penalty. Use training/validation sets to separately estimate the model and choose the hyperparameter (maximum clique size).

Topological regularization via persistence sensitive optimization A. Nigmatov et al. (2020)

Persistence holomogy can help control the complexity of the solution of an optimization problem, e.g., the decision boundary of a classifier, or the latent representation of an auto-encoder, by removing low-persistence 0-dimensional features (noise).

$$\text{Loss} = \sum_{d_i - b_i \leq \varepsilon} (d_i - b_i)^2$$

The 0-dimensional persistence diagram of a function $f : X \rightarrow \mathbf{R}$ is

$$\text{Dgm}(f) = \{(f(v_{\text{birth}}), f(v_{\text{death}}))\}.$$

An ε -simplification of f is a function g such that $\|f - g\|_\infty \leq \varepsilon$ and

$$\text{Dgm}(g) = \{(b, d) \in \text{Dgm } f : |d - b| > \varepsilon\}.$$

Persistence-sensitive optimization (PSO) alternates between two gradient descents (each with its own momentum):

- Gradient step to reduce $\text{Loss}(\theta)$;
- ε -simplification g of f_θ ;
- Gradient step to reduce $\|f_\theta - g\|^2$ (g fixed).

Second-order optimization made practical R. Anil et al. (2020)

Updates for gradient-based optimization are of the form $w \leftarrow w - \eta g$. Preconditioning methods are $w \leftarrow w - Pg$, where P is an $n \times n$ matrix computed from the Hessian (Newton) or gradient-gradient correlations. AdaGrad assumes P is diagonal, while *Shampoo* uses

$$\begin{aligned} W &\in \mathbf{R}^{m \times n} \\ L_t &= \sum_{s=1}^t G_s G_s' \in \mathbf{R}^{m \times m} \\ R_t &= \sum G_s' G_s \in \mathbf{R}^{n \times n} \\ w &\leftarrow w - \eta L^{-1/4} G R^{-1/4}. \end{aligned}$$

The condition number for L is bad ($\geq 10^7$): double precision computations are required.

Also check: K-FAC, GGT.

The non-convex Burer-Monteiro approach works on smooth semi-definite programs N. Boumal et al. (2018)

The semi-definite program

$$\begin{aligned} \text{Find} \quad & X \in \mathbf{R}^{n \times n} \\ \text{To minimize} \quad & \langle C, X \rangle \\ \text{Such that} \quad & AX = b \\ & X \succeq 0 \end{aligned}$$

can be reformulated, if X has rank p , as

$$\begin{aligned} \text{Find} \quad & Y \in \mathbf{R}^{n \times p} \\ \text{To minimize} \quad & \langle CY, Y \rangle \\ \text{Such that} \quad & AYY' = b. \end{aligned}$$

It is not convex but, under reasonable assumptions, first- and second-order optimality conditions are also sufficient.

Fairwashing explanations with off-manifold detergent C.J. Anders et al. (2020)

Most explanations of black-box methods (gradient, $x \odot \text{grad}$ (contribution), integrated gradient, layerwise relevance propagation) ignore the data manifold and can be manipulated: keep the model as is on the data manifold and alter it orthogonally – in practice, add the desired explanation to the loss function (it is Goohart's law).

Projecting the explanations tangentially to the manifold makes them more robust (use the k -nearest neighbours, or an autoencoder, to estimate the tangent space).

***NCVis: noise contrastive approach
for scalable visualization***
A. Artemenkov and M. Panov (2020)

Noise contrastive estimation (NCE) maximizes

$$\mathbb{E}_{x \sim \text{data}} \log \frac{p_{\theta}(x)}{p_{\theta}(x) + \nu p_{\text{noise}}(x)} + \nu \mathbb{E}_{x \sim \text{noise}} \log \frac{\nu p_{\text{noise}}(x)}{p_{\theta}(x) + \nu p_{\text{noise}}(x)}$$

Simplify t-SNE by:

- Replacing the Gaussian distributions in the input space with uniform distributions on the k nearest neighbours;
- Treating the normalization constant in the low-dimensional space as a parameter;
- Using NCE with, as noise distribution, the data distribution on edges with one end replaced at random.

Python code available. Also check UMAP (slower), FIt-SNE.

***Efficient algorithms for t-distributed stochastic
neighbourhood embedding***
G.C. Linderman et al. (2017)

FIt-SNE is a faster implementation of t-SNE (`fast_tsne` in R or Python).

It is also useful in dimension 1: plot it several times, with dodging, and colours from DBscan or features of interest.

***Arbitrary style transfer in real-time
with adaptive instance normalization***
X. Huang and S. Belongie (2017)

Instead of *training* a style transfer network so that the statistics of the latent representation match those of the target image (a time-consuming optimization), **AdaIN** (adaptive instance normalization) just *rescales* them.

***RealMix: towards realistic
semisupervised deep learning algorithms***
V. Nair et al. (2019)

Semi-supervised learning relies on a few ideas:

- Consistency training: outputs on unlabeled data should be invariant to small perturbations;
- Entropy minimization: decision boundaries should avoid dense regions (the model should be more confident);
- MixUp: linear interpolation of (labeled) data points;
- Data augmentation;
- Out-of-distribution masking: mask unlabeled samples with low confidence (to account for different distributions in the labeled and unlabeled sets);
- Training signal annealing: limit training on labeled samples (there are too few of them) to avoid overfitting.

***Inclusive GAN: improving data
and minority coverage in generative models***
N. Yu et al. (2020)

Implicit maximum likelihood estimation (IMLE) alleviates the mode collapse problem of GANs by matching each real sample with a generated one (instead of the opposite).

$$\text{Minimize}_{\theta} \mathbb{E}_{z_1, \dots, z_n \sim N(0, I)} \mathbb{E}_{x \sim \text{data}} \min_i \|G_{\theta}(z_i) - x\|^2$$

(use a fast nearest neighbour algorithm, e.g., prioritized DCI). One can combine adversarial and reconstructive (IMLE) losses.

***Fast k-nearest neighbour search
via prioritized DCI***
K. Li and J. Malik (2017)

Fast (approximate) nearest neighbour search can be done with k-d-trees (only in low dimension) or locality-sensitive hashing (LSH – but performance degrades if there are large density variations).

Dynamic continuous indexing (DCI) does not partition the points, but uses indices, each ordering the points in a random direction. *Prioritized DCI* does not visit one point from each index at each iteration, but prioritizes them (with the distance, along the projection direction, between the query and the next point).

***Finding nearest neighbours
in growth-restricted metrics***
D.R. Karger and I.M. Ruhl (2002)

To perform nearest neighbour search in an arbitrary metric space, store, for each observation, a random list of neighbours at distance $\leq r$, for several values of r , and use them, progressively reducing r , to move closer to the target point.

A discrete metric space has (ρ, c) -expansion if

$$\forall p \in X \quad |B_p(r)| \leq \rho \implies |B_p(2r)| \leq c |B_p(r)|.$$

For instance, (\mathbf{Z}^d, L^1) has $(1, 2^d)$ -expansion.

Applications if the KR dimension include internet routing and machine learning (the manifold hypothesis).

***JASS: Japanese-specific sequence to sequence
pretraining for neural machine translation***
Z. Mao et al. (2020)

Pretrain the model on units larger than words (文節 \approx noun or verb phrase) on a masked prediction or re-ordering task.

***Tapas: weakly supervised table parsing
via pre-training***
J. Herzig et al. (2020)

Train a BERT model for question answering on table data (free-form questions, asking which cells to select and which aggregation function (sum, avg, count, nop) to use) on data from Wikipedia (Infobox, Wikitable).

***Synthesizer: rethinking self-attention
for transformer models***
Y. Tay et al. (2020)

In the transformer $X \mapsto \text{Softmax}(QK')V$, replace the dot product QK' with:

- a shallow net $F : \mathbf{R}^d \rightarrow \mathbf{R}^\ell$, operating identically on each of the ℓ tokens $\mathbf{R}^{d \times \ell} \rightarrow \mathbf{R}^{\ell \times \ell}$
- or a random matrix.

SIGN: scalable inception graph neural network
F. Frasca et al. (2020)

Graph neural nets do not scale well to large graphs: one often uses seed nodes (picked at random, or using PageRank), adds their neighbourhoods, and only computes the loss on the seed nodes.

Instead, use a shallow net of the form

$$X \mapsto A_0 X \Theta_0, \dots, A_r X \Theta_r \xrightarrow{\sigma} Z \mapsto Z \Omega \xrightarrow{\sigma} \sigma(Z \Omega)$$

where Θ_k and Ω are parameters and the A'_k s are fixed operators (so that $A_k X$ can be precomputed), e.g., B^k , where B is the adjacency, the personalized pagerank adjacency, or triangle-based adjacency.

***Erdős goes neural:
an unsupervised learning framework
for combinatorial optimization on graphs***
N. Karalias and A. Loukas (2020)

To solve combinatorial optimization problems with deep learning, do not directly look for a solution, but for a *probability distribution* on candidate solutions: if $\mathbb{E}[f(S)] = a$, we know there is a solution S with $f(S) \leq a$.

For instance, given a graph G , to find a set of vertices $S \subset V(G)$ minimizing $f(S)$, for some nonnegative function f , use independent Bernoulli distributions on the vertices and $\text{Loss} = \mathbb{E}[f(S)]/(1-t)$: by the Markov inequality, it guarantees

$$P[f(S) \leq \text{loss}] \geq t.$$

To account for constraints $S \in \Omega$, add $\mathbf{1}_{S \in \Omega} \cdot \beta$ to f and find a computable differentiable upper bound for $P[S \in \Omega]$.

For box constraints

$$\sum_{v_i \in S} a_i \in [b_\ell, b_h],$$

rescale the probabilities to satisfy the constraint in expectation

$$p_i \leftarrow \text{clamp}(cp_i, 0, 1)$$

$$\sum_{v_i \in V} a_i p_i = \frac{1}{2}(b_\ell + b_h).$$

To retrieve integral solutions, use (sampling or) conditional expectations: sort the vertices by decreasing

probabilities and decide to add them to S one by one, by comparing

$$\mathbb{E}[f(S) \mid v_{i+1} \in S]$$

and $\mathbb{E}[f(S) \mid v_{i+1} \notin S, S_i \subset S]$.

Applications include max-clique and constrained min-cut.

***An alternative to backpropagation
in deep reinforcement learning***
S. Chung (2020)

If the mapping from state S to action A is given by a multi-layer network of *stochastic units*

$$H_\ell \sim P(H_\ell | H_{\ell-1}, W_\ell),$$

the REINFORCE gradient estimator

$$\nabla_W \mathbb{E} G = \mathbb{E}[G \nabla_W \log P(A|S)]$$

can be written

$$\nabla_{W_\ell} \mathbb{E} G = \mathbb{E}[G \nabla_{W_\ell} \log P(H_\ell | H_{\ell-1})]$$

which suggests the update rule

$$W_\ell \leftarrow W_\ell + \alpha G \nabla_{W_\ell} \log P(H_\ell | H_{\ell-1}).$$

The following has lower variance

$$W_\ell \leftarrow W_\ell + \alpha G \mathbb{E}[\nabla_{W_\ell} \log P(H_\ell | H_{\ell-1}) | S, A]$$

and the expectation can be approximated at the mode \hat{H}_ℓ of $P(H_\ell | S, A)$

$$W_\ell \leftarrow W_\ell + \alpha G \nabla_{W_\ell} \log(\hat{H}_\ell | \hat{H}_{\ell-1});$$

the modes can be estimated by gradient ascent.

***Realm: retrieval-augmented
language model pre-training***
K. Guu et al. (2020)

Augment a masked language model (BERT) with a retrieval system (maximum inner product search) and train them together.

Sampled softmax with random Fourier features
A.S. Rawat et al. (2020)

Since sampling from the full softmax distribution p is expensive

$$h : \text{input embedding}$$

$$c_i : \text{embedding of class } i$$

$$o_i = \tau h' c_i \text{ raw score (logit)}$$

$$p_i = e^{o_i} / Z \text{ probability}$$

$$t : \text{true label}$$

$$\mathcal{L}(x, t) : \text{loss}$$

$$\nabla_{\theta} \mathcal{L} = -\nabla_{\theta} o_t + \mathbb{E}_{s \sim p} [\nabla_{\theta} o_s]$$

it is often replaced by an easier distribution, e.g., uniform, or some prior – but the softmax distribution is

the only one which gives an unbiased estimator. To have a low bias, we need the p_i/q_i to be close to 1.

Kernel-based sampling uses $q_i \propto k(h, c_i)$, where $k : \mathbf{R}^d \times \mathbf{R}^d \rightarrow \mathbf{R}$ is some kernel. If the kernel can be linearized, $k(h, c) = \phi(h)' \phi(c)$, for $\phi : \mathbf{R}^d \rightarrow \mathbf{R}^D$, sampling can be done in $O(D \log n)$ by a divide-and-conquer algorithm (put the classes in a binary tree: we only need $\sum_{c \in S} \phi(c)$ for the subtrees S).

The softmax is obtained for $k(h, c) = \exp(\tau h'c)$. The quadratic kernel

$$k(h, c) = \alpha(h'c)^2 + 1$$

$$\phi(x) = [\sqrt{\alpha}(z \otimes z), 1]$$

is not a good approximation of the exponential kernel, and does not scale well with the embedding dimension d .

Use *random Fourier features* (and a normalized embedding)

$$\phi(u) = \frac{1}{\sqrt{D}} [\cos w'_1 u, \dots, \cos w'_D u, \sin w'_1 u, \dots, \sin w'_D u]$$

$$w_i \sim N(1, 1/\tau)$$

$$\exp -\frac{\tau}{2} \|x - y\|^2 \approx \phi(x)' \phi(y).$$

Transformers are RNNs: fast autoregressive transformers with linear attention
A. Katharopoulos et al. (2020)

The softmax in the attention mechanism can be linearized, e.g., with a quadratic kernel, random Fourier features, or $\phi(x) = \text{elu}(x) + 1$.

Language ID in the wild: unexpected challenges on the path to a thousand-language web text corpus
I. Caswell et al. (2020)

Language identification on real (dirty, web-crawled) data for low-resource languages is tricky: avoid overly cleaned data, deduplicate, balance the data, use curated wordlists (500/language), train a transformer and check the types or errors (mojibake, create use of Unicode, obfuscation boilerplate, high-resource nearly language, n -gram overlap, antspeak, repetitions, etc.)

A convenient generalization of Schlick's bias and gain functions
J.T. Barron (2020)

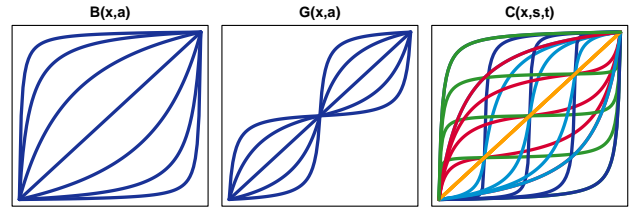
Increasing functions $[0, 1] \rightarrow [0, 1]$ with at most one

inflection point $(a, t \in (0, 1), s \geq 0)$.

$$B(x, a) = \frac{x}{(1/a - 2)(1 - x) + 1}$$

$$G(x, a) = \begin{cases} \frac{B(2x, a)}{2} & \text{if } x < 1/2 \\ \frac{B(2x - 1, 1 - a) + 1}{2} & \text{if } x \geq 1/2 \end{cases}$$

$$C(x, s, t) = \begin{cases} \frac{tx}{x + s(t - x) + \varepsilon} & \text{if } x < t \\ \frac{(1 - t)(x - 1)}{1 - x - s(t - x) + \varepsilon} + 1 & \text{if } x \geq t \end{cases}$$



***q*-neurons: neuron activations based on stochastic Jackson's derivative operators**
F. Nielsen and K. Sun (2018)

Replace the activation function f with

$$g_q(x) = \frac{f(x) - f(qx)}{1 - q},$$

with $q \sim N(1, \sigma^2)$ (but remove a small interval around 1 for stability). The gradient of g contains second-order information:

$$\lim_{\text{Var } q \rightarrow 0} g'_q(x) = f'(x) + f''(x)x.$$

The q - and (p, q) -derivatives (from quantum calculus) are

$$D_q f(x) = \frac{f(x) - f(qx)}{(1 - q)x}$$

$$D_{p,q} f(x) = \frac{f(px) - f(qx)}{(p - q)x}.$$

Learning with differentiable perturbed optimizers
Q. Berthet et al. (2020)

To differentiate through an Argmax

$$C = \text{Conv}\{y_1, \dots, y_n\}$$

$$F(\theta) = \text{Max}_{y \in \{y_1, \dots, y_n\}} \langle y, \theta \rangle = \text{Max}_{y \in C} \langle y, \theta \rangle$$

$$y^*(\theta) = \text{Argmax}_{y \in C} \langle y, \theta \rangle = \nabla_{\theta} \text{Max}_{y \in C} \langle y, \theta \rangle$$

add noise $z \sim \exp -U(z)$ (this generalizes the Gumbel softmax trick, for which $C = \Delta$ is the standard simplex).

$$F_{\varepsilon}(\theta) = \mathbb{E}_z [F(\theta) + \varepsilon Z]$$

$$y_{\varepsilon}^*(\theta) = \mathbb{E}_z [\text{Argmax}_y \langle y, \theta + \varepsilon Z \rangle] = \nabla_{\theta} F_{\varepsilon}(\theta)$$

The Jacobian can be computed with Monte Carlo simulations.

$$y_\varepsilon^*(\theta) = \mathbb{E}[F(\theta + \varepsilon Z) \nabla_Z \nu(Z) / \varepsilon]$$

$$J_\theta y_\varepsilon^*(\theta) = \mathbb{E}[F(\theta + \varepsilon Z) \nabla_X \nu(Z) \nabla_Z \nu(Z)' - \nabla_Z \nu(Z)^2 / \varepsilon^2]$$

Applications include rank correlation, shortest path.

Geometric distances via optimal transport
D. Alvarez-Melis and N. Fusi (2020)

Optimal transport on $(\mathcal{X}, \mathcal{Y})$ can be used to compare datasets (e.g., to choose the closest one for transfer learning). If the labels are not the same, replace them with the corresponding distribution in feature space (input), and use the optimal transport distance between those distributions as “distance” (transportation cost) between the labels.

Lambda networks: modeling long-range interactions without attention
(2020)

Lambda layers are a variant of / an alternative to attention layers, which map a context to a linear function, which is then applied to the input (keys).

Fourier neural operator for parametric partial differential equations
Z. Li et al. (2020)

An infinite-dimensional generalization of neural nets approximates a function

$$G : A = \mathcal{F}(D, \mathbf{R}^{d_a}) \longrightarrow U = \mathcal{F}(D, \mathbf{R}^{d_u})$$

by minimizing $\mathbb{E}_{a \sim \mu} \text{Cost}(G_\theta(a), G(a))$. First, lift the input to a higher-dimensional representation, with a shallow neural net

$$\begin{aligned} v_0 : D &\longrightarrow \mathbf{R}^{d_a} \\ v_1 : D &\xrightarrow{v_0} \mathbf{R}^{d_a} \xrightarrow{P} \mathbf{R}^{d_v}, \end{aligned}$$

then apply iterative updates

$$v_{t+1}(x) = \sigma \left(W v_t(x) + \int_D k_\phi(x-y) v_t(y) dy \right);$$

parametrize $k_\phi : \mathbf{R}^{d_a} \longrightarrow \mathbf{R}^{d_v \times d_v}$ in Fourier space and use the FFT to compute the convolution.

Algebraic statistics in practice
M. Casanellas et al. (2019)

Exponential random graph models (ERGM) are of the form

$$p_\theta(G) \propto \exp\langle T(G), \theta \rangle.$$

The *fiber* of G is the set of graphs with the same sufficient statistic $\{G' : T(G') = T(G)\}$. For log-linear models, *i.e.*, $T(G) = BA_G$, where A_G is the adjacency matrix, *toric ideals* can help describe the fiber and sample from it (Metropolis-Hastings), to perform goodness-of-fit tests.

The number of Markov equivalence classes (MEC) of directed graphical models on n nodes, and the number of MECs with only one GDAG, are open problems in combinatorics.

To infer the structure of a directed graphical model with conditional independence tests, it has to be *faithful*, *ie*, causal effects should not cancel out: (for Gaussian linear structural equation models (SEM)), that condition defines a real algebraic surface, and the volume of tubes around it tells us how many samples are needed.

Instead of testing all conditional independence relations, choose an order on the vertices and only test for $X_i \perp\!\!\!\perp X_j \mid X_{\leq j, \neq i, j}$, $i < j$. Prefer the order leading to the sparsest DAG (sparsest permutation search – a vertex on the permutahedron, polytope whose vertices are orderings of $\llbracket 1, n \rrbracket$ and with edges when two permutations differ by a transposition of adjacent elements). (Things get even more complicated with unobserved variables.)

Phylogenetics models consider a probability distribution on $\{A, C, T, G\}^n$ at the root of the tree (e.g., uniform), a transition matrix for each edge (e.g., with a single parameter, $P(x|y)$ for $x \neq y$), and we observe the probabilities on the leaves. Some relations between the probabilities on the leaves (the fact that some matrix has low rank) depend on the structure of the tree and the model: they can help identify the tree, select the model, or prove model identifiability.

Factor investing with Black-Litterman-Bayes: incorporating factor views and priors in portfolio construction
P.N. Kolm and G. Ritter (2020)

1. The simplified Black-Litterman framework models stock returns as

$$\begin{aligned} r &\sim N(\mu, \Sigma) && \text{stock returns} \\ Z &\sim N(0,) && \text{views uncertainty} \\ Q &= Pr + Z && \text{views} \end{aligned}$$

and then computes the conditional distribution of returns given the views, $r \mid Q = q$.

2. The textbook Black-Litterman model is more complicated: the views are not on the returns but on the expected returns.

$$\begin{aligned} \mu &\sim N(\pi, C) && \text{expected stock returns} \\ r &\sim N(\mu, \Sigma) && \text{stock returns} \\ Z &\sim N(0, \Omega) && \text{uncertainty of the views} \\ Q &= P\mu + Z && \text{views on expected returns} \end{aligned}$$

We can then compute the conditional distributions $\mu \mid Q = q$ and $r \mid Q = q$, and the optimal (unconstrained) portfolio

$$w^* = \underset{w}{\text{Argmax}} w' \mu - \frac{1}{2} \lambda w' \Sigma w = \lambda^{-1} \Sigma^{-1} \mu.$$

3. The *factor Black-Litterman* replaces the stock return model, $r \sim N(\mu, \Sigma)$, with a factor model, $r = Xf + \varepsilon$; the views are on the expected factor returns μ (not the factor returns).

$$\begin{aligned}\mu &\sim N(\xi, V) && \text{expected factor returns} \\ f &\sim N(\mu, F) && \text{factor returns} \\ r &= Xf + \varepsilon && \text{factor model} \\ \varepsilon &\sim N(0, D) \\ Z &\sim N(0, \Omega) && \text{uncertainty of the views} \\ Q &= \mu + Z && \text{views on expected factor returns}\end{aligned}$$

From this model, we can compute the conditional distributions of the expected factor returns $\mu | Q = q$, then of the factor returns $f | Q = q$, then of the stock returns $r | Q = q$.

***Portfolio optimization
with drawdown constraints***
A. Chekhlov et al. (2003)

The *conditional drawdown* (or conditional drawdown at risk, or conditional expected drawdown) is the average of the worst $\alpha\%$ drawdowns. Average and maximum drawdowns are special cases ($\alpha = 1$ or 0). For portfolio optimization, assume the portfolio weights are constant and consider “uncompounded” returns (*i.e.*, assume that log- and ratio-returns are the same).

***Drawdown:
from practice to theory and back again***
L.R. Goldberg and O.Mahmoud (2014)

Deep fundamental factor models
M.F. Dixon and N.G. Polson (2019)

Use a neural network to forecast 1-period ahead stock returns (Russell 1000) from stock-level investment factors,

$$y_{t+1} = f(x_t) + \text{noise}.$$

The gradient (and the Hessian, for interactions) makes the model interpretable. [Data and code available]

***A study on the use of artificial intelligence
for learning characteristics of fund's behavior***
T. Tajiri (2020)

Dimension reduction (VAE) from holdings data.

A basket half-full: sparse portfolios
E. Seregina (2020)

Nodewise regression estimates the precision matrix (inverse of the variance) of x_1, \dots, x_n with a regression for each column, $\beta_{ij} = \text{coef}(x_i \sim s_{-i})$, as

$$C = \begin{pmatrix} 1 & -\beta_{12} & \dots & \dots \\ -\beta_{21} & \ddots & \ddots & \ddots \\ \vdots & \ddots & \ddots & \ddots \\ \vdots & \vdots & \vdots & 1 \end{pmatrix}$$

with a lasso penalty to encourage sparsity, rescaled with the variance of the residuals $\Theta = \Sigma^{-2}C$, $\Sigma = \text{diag}(\sigma_1^2, \dots, \sigma_n^2)$; symmetrize (with $\text{Min}(C, C')$) and discard negative eigenvalues. If there are many variables, first use a low-rank factorization $X = BF + E$, then sparse nodewise regression on the residuals, and finally Woodbury's formula to recover the precision of X .

***When bots take over the stock market:
evasion attacks against algorithmic traders***
E. Nehemya et al. (2020)

To build an adversarial attack on an automatic trading algorithm f , find a perturbation v such that $f(x+v) = y$ with high probability, where x is the input and y the desired output (we do not know $f(x)$ in advance: it could already be y). It is a white-box attack, but it may transfer.

Quant bust 2020
Z. Kakushadze (2020)

Quant strategies such as stat-arb and, more generally, medium-term strategies (1 day to 1 month) rely on long-term investors keeping their positions and ignoring potentially profitable medium-term fluctuations. During the Covid-19 crisis, they sold their positions: that assumption was broken.

***Deep reinforcement learning
for asset allocation in US equities***
M. Alonso and S. Srivastava (2020)

CNN, RNN, LSTM, from 50 years of HLC prices for 25 stocks to maximize the average log-returns; the generated weights are combined with the previous 20 ones with a 1×1 convolution to reduce turnover (this drastically reduces performance as well).

***Explainable machine learning
in credit risk management***
N. Bussmann et al. (2020)

Use the correlation between the Shapley contributions to group the predictors.

An operational measure of riskiness
D.P. Foster and S. Hart (2009)

Given a discrete random variable X modeling the outcome of a gamble, *i.e.*, such that $EX > 0$ and $P[X < 0] > 0$, its Foster-Hart risk r is the positive solution of

$$E \log \left(1 + \frac{X}{r} \right) = 0;$$

it is greater than the maximum loss and guarantees no-bankruptcy. For continuous distributions, it need not be defined: use the maximum loss instead.

Parametric UMAP: learning embeddings with deep neural networks for representation and semi-supervised learning
T. Sainburg et al. (2020)

UMAP learns an embedding of the data points that preserves the graph structure of the data; parametric UMAP uses a neural network to compute that embedding (instead of looking for the coordinates of each point in the dataset), which allows adding new points. Contrary to t-SNE, the loss function uses negative sampling and does not require normalization.

$$\begin{aligned}
 p_{j|i}^{\text{t-SNE}} &\propto \exp - \frac{d(x_i, x_j)}{2\sigma_i^2} \\
 p_{j|i}^{\text{UMAP}} &= \exp - \frac{d(x_i, x_j) - \rho_i}{\sigma_i} \\
 q_{ij}^{\text{t-SNE}} &\propto (1 + \|z_i - z_j\|^2)^{-1} \\
 q_{ij}^{\text{UMAP}} &= (1 + a \|z_i - z_j\|^{2b})^{-1} \\
 \text{Loss}^{\text{t-SNE}} &= \text{KL}(p, q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \\
 \text{Loss}^{\text{UMAP}} &= \text{CE}(p, q) \\
 &= \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}} + (1 - p_{ij}) \log \frac{1 - p_{ij}}{1 - q_{ij}}
 \end{aligned}$$

The UMAP loss can be used as a regularization to learn a latent representation preserving the structure of the data, in autoencoders or for semi-supervised learning.

From trees to continuous embeddings and back: hyperbolic hierarchical clustering
I. Chami et al. (2020)

Hierarchical clustering algorithms are often heuristics. More rigorously, one can minimize the DasGupta cost

$$\begin{aligned}
 \text{Cost}(T) &= \sum_{ij} w_{ij} |\text{leaves } T(i \vee j)| \\
 &= \sum_{ijk} (w_{ij} + w_{ik} + w_{jk} - w_{ijk}) + 2 \sum_{ij} w_{ij}
 \end{aligned}$$

where

$$\begin{aligned}
 i, j: & \text{leaves} \\
 i \vee j: & \text{lowest common ancestor (LCA)} \\
 T(i \vee j): & \text{tree below } i \vee j \\
 w_{ij}: & \text{similarity of } i \text{ and } j \\
 w_{ijk} &= \begin{pmatrix} w_{ij} \\ w_{ik} \\ w_{jk} \end{pmatrix} \cdot \begin{pmatrix} \mathbf{1}_{\{i,j|k\}} \\ \mathbf{1}_{\{i,k|j\}} \\ \mathbf{1}_{\{j,k|i\}} \end{pmatrix} \\
 \{i, j|k\}: & i \vee j \text{ is a descendant of } i \vee j \vee k.
 \end{aligned}$$

This combinatorial optimization problem can be relaxed by looking for a hyperbolic embedding, replacing $i \vee j$ with the point on the geodesic between i and j closest to the origin, and

$$\begin{pmatrix} \mathbf{1}_{\{i,j|k\}} \\ \mathbf{1}_{\{i,k|j\}} \\ \mathbf{1}_{\{j,k|i\}} \end{pmatrix} \quad \text{with} \quad \text{softmax}_{\tau} \begin{pmatrix} d(i \vee j, 0) \\ d(i \vee k, 0) \\ d(j \vee k, 0) \end{pmatrix}.$$

To reconstruct the tree, use $d(i \vee j, 0)$. (To speed up the $O(n^3)$ sum, use all (i, j) pairs, but sample a single k at random.)

This can be used as a (differentiable) step in a longer ML pipeline (end-to-end training).

AlgebraNets
J. Hoffmann et al. (2020)

Try using algebras other than \mathbf{R} for weights and activations (\mathbf{C} , \mathbf{H} , $\mathbf{C}^{2 \times 2}$, $\mathbf{R}^{2 \times 2}$, $\mathbf{R}^{3 \times 3}$, $\mathbf{R}^{4 \times 4}$, dual numbers $\mathbf{R}[\varepsilon]/\langle \varepsilon^2 \rangle$, cross-product \mathbf{R}^3): often, fewer parameters suffice.

Fast and simple PCA via convex optimization
D. Garber and E. Hazan (2015)

To find the largest eigenvalue of a positive definite matrix, use the power method on $M^{-1} = (\lambda I - X)^{-1}$ for $\lambda > \lambda_1(X)$: it has the same eigenvectors, but its condition number $\lambda_1/(\lambda_1 - \lambda_2)$ can be lower. The product $M^{-1}w$ required by the power method can be efficiently computed by solving a convex problem, $M^{-1}w = \text{Argmin}_z \frac{1}{2} z' M z - w' z$.

Use variance-reduced stochastic gradient (SVRG) and progressively reduce λ to bring it closer to $\lambda_1(X)$.

Sharpness minimization for efficiently improving generalization
P. Foret et al. (2020)

Simultaneously minimize loss and loss sharpness by replacing $\text{Loss}(w) + \lambda \|w\|_2^2$ with

$$\text{Max}_{\|\varepsilon\|_p \leq \rho} \text{Loss}(w + \varepsilon) + \lambda \|w\|_2^2.$$

Use the first-order expansion of the loss to approximate the solution of the inner maximization problem and compute the gradient). Only compute the sharpness, $\text{Max}_{\varepsilon} \text{Loss}(w + \varepsilon) - \text{Loss}(w)$, on minibatches.

Fact or fiction: verifying scientific claims
D. Wadden et al. (2020)

Given a statement, use BERT (SciBERT, BioMet ROBERTa) to label sentences from paper abstracts as supports/refutes/no_info.

Automatic generation of reviews of scientific papers
A. Nikiforovskaya et al. (2020)

Combine bibliometric analysis (PageRank and Louvain community detection on the citation, co-citation, or text similarity graphs) with extractive summarization (BERTSUM computes the relevance of each sentence for a summary) to generate a review “paper” for a domain, i.e., identify and summarize the (many) relevant papers in a domain (the output is a table with columns

“paper”, “summary”, “relevance”). The *citation context* (the sentence citing a paper) provides additional context.

***Descending through a crowded valley:
benchmarking deep learning optimizers***
R.M. Schmidt et al. (2020)

The best optimizer is problem-dependent, but:

- Adam, RAdam, NAdam, AMSBound, AdaBound have robust default parameters;
- A non-constant learning rate schedule may help: cosine λ , cosine with restarts $\lambda \lambda \lambda$, or trapezoidal λ .

***Beyond vector spaces:
compact data representation
as differentiable weighted graphs***
D. Mazur et al. (2019)

To account for arbitrary geometries, embed your data in a graph rather than a vector space [PyTorch implementation available].

$p_\theta(i, j)$ edge probability

$w_\theta(i, j)$ edge weight

$$d(i, j) = \min_{\pi: i \rightarrow j} \sum_{e \in \pi} w_\theta(e)$$

$$\text{Loss} = \mathbb{E}_{G \sim p_\theta} L(G) + \frac{1}{|E|} \sum_{ij} p_\theta(i, j)$$

(use the “log-derivative trick” (REINFORCE) for the gradient). After optimization, the edge probabilities are close to 0 or 1: discard all those below 0.5 to have a sparser graph. Examples include generalization of MDS (or Poincaré MDS) and collaborative filtering.

$$L(G) = (\|x_i - x_j\| - d(i, j))^2$$

$$L(G) = \left[1 + \frac{e^{-d(u_i, x^+)}}{e^{-d(u_i, x^-)}} \right]^{-1}$$

***Embedding words in non-vector spaces
with unsupervised graph learning***
M. Ryabinin et al. (2020)

Embedding words (GloVe) in Euclidean or hyperbolic space (constant curvature) assumes the geometry of the space is the same everywhere: instead, use a graph embedding.

Poincaré GloVe: hyperbolic word embedding
A. Tifrea et al. (2019)

The *Gromov δ -hyperbolicity* measures how tree-like (hyperbolic) a metric space is: it is half (the average of) the difference between the biggest two of $d(x, y) + d(z, t)$, $d(x, z) + d(y, t)$, $d(x, t) + d(y, z)$ (normalize it by dividing by the average distance; hyperbolic spaces have lower values).

Words can be embedded as Gaussian distributions – but the space of Gaussian distributions is hyperbolic –

the space of diagonal Gaussians is isometric to a product of Poincaré half planes $(\mathbb{H}^2)^n$.

***Gradient estimation
with stochastic softmax tricks***
M.B. Paulus et al. (2020)

Sampling from a discrete distribution

$$U \sim U(0, 1)$$

$$i = \text{Max}\{i : p_1 + \dots + p_{i-1} \leq U\}$$

$$Z = \text{onehot}(i)$$

can be parametrized with a Gumbel distribution

$$G_i \sim \text{Gumbel}(0, 1)$$

$$i = \text{Argmax}(G_i + \log p_i)$$

$$Z = \text{onehot}(i)$$

and relaxed (*Gumbel softmax*)

$$G_i \sim \text{Gumbel}(0, 1)$$

$$X_i = G_i + \log p_i$$

$$Z = \text{softmax}(X).$$

The “reparametrization trick” is used to differentiate wrt the parameters of a discrete distribution (in the backward pass – in the forward pass, you may still use discrete distributions). It can also be written

$$Z = \text{Argmax}_{x \in \text{conv } \mathcal{X}} U'x - tf(x)$$

$$U \sim \text{Gumbel}$$

$$\mathcal{X} = \Delta_n$$

$$f(x) = \sum x_i \log x_i$$

t : temperature

Deep Bayes 2019

1. Frequentist and Bayesian approaches do not disagree: maximum likelihood only provides asymptotic guarantees, and is asymptotically equivalent to the Bayesian approach.

Frequentist and Bayesian approaches use probabilities to encode different types of randomness. The frequentist approach assumes randomness in the data generation process – even though there is rarely any actual physical (e.g., quantum) randomness, just complex, chaotic phenomena that look random. The Bayesian approach uses probability theory to model uncertainty.

Likelihood	Parameters	Conjugate prior
Gaussian	μ	Gaussian
	σ^{-2}	Gamma
	(μ, σ^{-2})	Gaussian-Gamma
	Σ^{-1}	Wishart
Bernoulli	p	Beta
Multinomial	(p_1, \dots, p_m)	Dirichlet
Poisson	λ	Gamma
Uniform	θ	Pareto

2. The posterior distribution $p(\theta|x)$ is often intractable: **variational inference** replaces it with a simpler distribution $q(\theta)$, as close as possible for the (mode-seeking) Kullback-Leibler divergence.

$$q = \underset{q}{\operatorname{Argmax}} \operatorname{KL}(q(\theta) \parallel p(\theta|x))$$

$$\operatorname{KL}(q \parallel p) = \int \log \frac{q(\theta)}{p(\theta)} q(\theta) d\theta$$

Since the posterior is intractable, so seems the KL divergence.

However, the *evidence* $p(x)$ can be decomposed as

$$\begin{aligned} \log p(x) &= \int \log p(x) q(\theta) d\theta \\ &= \int \log \frac{p(x, \theta)}{p(\theta|x)} q(\theta) d\theta \\ &= \int \log \left(\frac{p(x, \theta)}{p(\theta|x)} \frac{q(\theta)}{q(\theta)} \right) q(\theta) d\theta \\ &= \int \log \frac{p(x, \theta)}{q(\theta)} q(\theta) d\theta + \int \log \frac{q(\theta)}{p(\theta|x)} q(\theta) d\theta \\ &= \mathcal{L}(q(\theta)) + \operatorname{KL}(q(\theta) \parallel p(\theta|x)). \end{aligned}$$

Since the lhs does not depend on q , to minimize the KL divergence, it suffices to maximize the **evidence lower bound** (ELBO) $\mathcal{L}(q(\theta))$.

Note that, since $\operatorname{KL} \geq 0$, it is indeed a lower bound: $\log p(x) \geq \mathcal{L}(q(\theta))$.

The ELBO can be decomposed as

$$\begin{aligned} \mathcal{L}(q(\theta)) &= \int \log \frac{p(x, \theta)}{q(\theta)} q(\theta) d\theta \\ &= \underset{\theta \sim q}{\operatorname{E}} [\log p(x|\theta)] - \operatorname{KL}(q(\theta) \parallel p(\theta)). \end{aligned}$$

The first term, the expected log-likelihood, is maximum if $q = \delta_{\hat{\theta}_{\text{MLE}}}$; the second term, the KL divergence with the prior $p(\theta)$, is minimum if $q = p$, and acts as a regularizer. This decomposition is similar to the Bayes formula:

$$\begin{aligned} \text{Posterior} &\propto \text{likelihood} \times \text{prior} \\ \text{ELBO} &= \text{data term} + \text{regularizer}. \end{aligned}$$

The *mean-field approximation* constrains q to be of the form $q(\theta) = \prod_i q_i(\theta_i)$. *Block coordinate descent* updates the estimate of q_j while keeping the other q_i 's

fixed.

$$\begin{aligned} \mathcal{L}(q(\theta)) &= \int \log \frac{p(x, \theta)}{q(\theta)} q(\theta) d\theta \\ &= \underset{\theta \sim q}{\operatorname{E}} \log p(x, \theta) - \underset{\theta \sim q}{\operatorname{E}} \log q(\theta) \\ \underset{\theta \sim q}{\operatorname{E}} \log p(x, \theta) &= \underset{\theta_j \sim q_j}{\operatorname{E}} \underset{\substack{\theta_i \sim q_i \\ i \neq j}}{\operatorname{E}} \log p(x, \theta) \\ \underset{\theta \sim q}{\operatorname{E}} \log q(\theta) &= \sum_i \underset{\theta_i \sim q_i}{\operatorname{E}} \log q_i(\theta_i) \\ &= \underset{\theta_j \sim q_j}{\operatorname{E}} \log q_j(\theta_j) + \text{const} \\ \mathcal{L}(q(\theta)) &= \underset{\theta_j \sim q_j}{\operatorname{E}} \left[\underset{\substack{\theta_i \sim q_i \\ i \neq j}}{\operatorname{E}} \log p(x, \theta) - \log q_j(\theta_j) \right] + \text{const} \\ &= \underset{\theta_j \sim q_j}{\operatorname{E}} [\log r_j(\theta_j) - \log q_j(\theta_j)] + \text{const} \\ &= -\operatorname{KL}(q_j(\theta_j) \parallel r_j(\theta_j)) + \text{const} \end{aligned}$$

This is maximum for $q_j = r_j$; it can be computed if prior and likelihood are conditionally conjugate, *i.e.*, when $p(\theta_j|\theta_{i \neq j})$ and $p(\theta_j|x, \theta_{i \neq j})$ are in the same family.

3. To estimate a mixture of Gaussians

$$\begin{aligned} Z_i &\sim \text{Categorical}(\pi_1, \dots, \pi_k) \\ X_i &\sim N(\mu_{z_i}, \sigma_{z_i}^2) \\ \theta_j &= (\mu_j, \sigma_j^2), \end{aligned}$$

use the evidence decomposition

$$\begin{aligned} \log p(X|\theta) &= \int \log \frac{p(X, Z|\theta)}{q(Z)} q(Z) dZ + \\ &\quad \int \log \frac{q(Z)}{p(Z, X, \theta)} q(Z) dZ \\ &= \mathcal{L}(q, \theta) + \operatorname{KL}(q(Z) \parallel p(Z|X, \theta)) \\ &\geq \mathcal{L}(q, \theta) \end{aligned}$$

Insead of maximizing $p(X|\theta)$, maximize $\mathcal{L}(q, \theta)$, wrt q and θ . The *EM algorithm* is the block-coordinate ascent algorithm to maximize \mathcal{L} .

$$\begin{aligned} \text{E step} \quad q &= \underset{q}{\operatorname{Argmax}} \mathcal{L}(q, \theta) \\ &= \underset{q}{\operatorname{Argmin}} \operatorname{KL}(q(Z) \parallel p(Z|X, \theta)) \\ &= p(Z|X, \theta) \\ \text{M step} \quad \theta &= \underset{\theta}{\operatorname{Argmax}} \mathcal{L}(q, \theta) \\ &= \underset{\theta}{\operatorname{Argmax}} \underset{Z \sim q}{\operatorname{E}} \log p(X, Z|\theta) \end{aligned}$$

If Z is discrete, this is tractable.

$$\begin{aligned} q(Z_i = k) &= \frac{p(x_i|Z_i = k, \theta)p(Z_i = k|\theta)}{\sum_{\ell} p(x_i|Z_i = \ell, \theta)p(Z_i = \ell|\theta)} \\ \theta &= \underset{\theta}{\operatorname{Argmax}} \sum_i \sum_k q(Z = k) \log p(x_i, Z = k|\theta) \end{aligned}$$

Variational EM uses variational inference in the E step.

A variational lower bound of f satisfies

$$\begin{aligned}\forall x \forall \xi \quad f(x) &\geq g(\xi, x) \\ \forall x \exists \xi \quad f(x) &= g(\xi, x)\end{aligned}$$

The MM algorithm iterates

$$\begin{aligned}x &\leftarrow \underset{x}{\operatorname{Argmax}} g(\xi, x) \\ \xi &\leftarrow \underset{\xi}{\operatorname{Argmax}} g(\xi, x)\end{aligned}$$

The *Ada-gram* (adaptive skipgram) algorithm extends the word2vec model by considering several embeddings for each word, corresponding to different meanings. Add a latent variable to select the meaning; use a Dirichlet process to have an unbounded number of meanings per word.

4. PCA is a latent variable model

$$\begin{aligned}p(X, Z|\theta) &= \prod_i p(x_i|z_i, \theta) p(z_i|\theta) \\ &= \prod_i N(x_i|Vz_i + \mu, \sigma^2 I) N(z_i|0, I);\end{aligned}$$

it can be estimated with the EM algorithm, generalized to *mixture PCA*

$$p(X, Z, Y|\theta) = \prod_i N(x_i|V_{t_i}z_i + \mu_{t_i}, \sigma_{t_i}^2 I) N(z_i|0, I) \pi_{t_i}$$

or non-linear PCA

$$\begin{aligned}p(X, Z|\theta) &= \prod_i p(x_i|z_i, \theta) p(z_i|\theta) \\ &= \prod_i \left[\prod_j N(x_{ij}|\mu_j(z_j), \sigma_j^2(x_j)) \right] N(z_i|0, I)\end{aligned}$$

(mean and variance of the factorized (diagonal) Gaussian are nonlinear functions of the latent variables instead of linear or constant).

For the E step, we need $q(z) = \prod_i p(z_i|x_i, \theta)$, which is intractable: use a variational approximation

$$\begin{aligned}q(z_i|x_i, \phi) &\approx p(z_i|x_i, \theta) \\ q(z_i x_i, \phi) &= \prod_j N(z_{ij}|\mu_j(x_i), \sigma_j^2(x_i))\end{aligned}$$

where μ_j and σ_j are other neural nets. The ELBO

$$\mathcal{L}(\phi, \theta) = \int q(Z|X, \phi) \log \frac{p(X|Z, \theta)p(Z)}{q(Z|X, \phi)} dZ$$

is intractable:

- Instead of doing a full optimization at each step, alternatively for ϕ and θ , optimize jointly for (ϕ, θ) using gradient descent;
- Use mini-batches;
- Express $\nabla_{\theta} \mathcal{L}$ and $\nabla_{\phi} \mathcal{L}$ as expectations, and approximate them with Monte Carlo;
- Somehow reduce the variance of those gradient estimators.

$$\nabla_{\theta} \mathbb{E}_{z \sim q_{\phi}} [f(z, \theta)] = \mathbb{E}_{z \sim q} [\nabla_{\theta} f(z, \theta)]$$

$$\nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}} [f(z, \phi)] = \mathbb{E}_{z \sim q_{\phi}} [\nabla_{\phi} f(z, \phi)] + \mathbb{E}_{z \sim q_{\phi}} [f(z, \phi) \nabla_{\phi} \log q_{\phi}(z)]$$

Monte Carlo estimators of the second term (REINFORCE estimator) are unstable: the score function, $\nabla_{\phi} \log q_{\phi}(z)$, whose expectation is zero, oscillates. To reduce the variance, use a baseline,

$$\mathbb{E}_{z \sim q_{\phi}} [(f(z, \phi) - B(\phi)) \nabla_{\phi} \log q_{\phi}(z)]$$

with $B(\phi) \approx E[f]$, or use the *reparametrization trick* (only for continuous variables), replacing $z \sim q_{\phi}$ with $\varepsilon \sim r, z = g_{\phi}(\varepsilon)$.

$$\begin{aligned}\nabla_{\phi} \mathbb{E}_{z \sim q_{\phi}} [f(z, \phi)] &= \nabla_{\phi} \mathbb{E}_{\varepsilon \sim r} [f(g_{\phi}(\varepsilon), \phi)] \\ &= \mathbb{E}_{\varepsilon \sim r} [\nabla_{\phi} f(g_{\phi}(\varepsilon), \phi)]\end{aligned}$$

q	r	$y = g_{\phi}(\varepsilon)$
$N(\mu, \sigma^2)$	$N(0, 1)$	$y = \mu + \sigma \varepsilon$
$\text{Gamma}(1, \beta)$	$\text{Gamma}(0, 1)$	$y = \beta \varepsilon$
$\text{Exp}(\lambda)$	$U(0, 1)$	$y = -\lambda^{-1} \log \varepsilon$
$N(\mu, AA')$	$N(0, I)$	$y = \mu + A \varepsilon$

A meta-transfer objective for learning to disentangle causal mechanisms Y. Bengio et al. (2019)

Latent representations that best encode causal relations are better at transfer learning: optimize for fast transfer learning and robustness to distributional changes.

For instance the following two models, for discrete variables, are equivalent,

$$\begin{aligned}P(A, B) &= P(A)P(B|A) \\ P(A, B) &= P(B)P(A|B)\end{aligned}$$

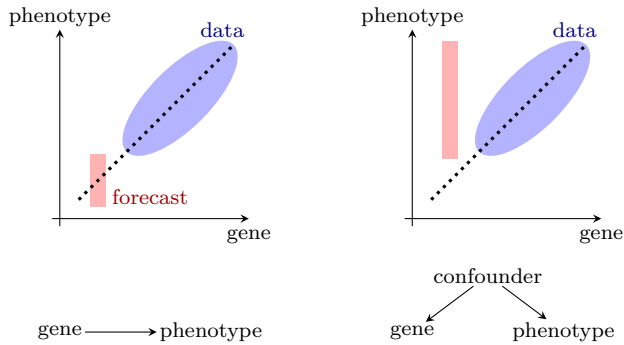
but if one undergoes a sparse change (e.g., an unknown intervention in a correct causal model), it will adapt faster.

Instead of considering separate models, one can consider a mixture.

$$\begin{aligned}R &= \log [\sigma(\gamma) \cdot \text{Lik}_{A \rightarrow B} + (1 - \sigma(\gamma)) \cdot \text{Lik}_{B \rightarrow A}] \\ \frac{\partial R}{\partial \gamma} &= \sigma(\gamma) - \sigma(\gamma + \Delta) \\ \Delta &= \log \frac{\text{Lik}_{A \rightarrow B}(\text{new data})}{\text{Lik}_{B \rightarrow A}(\text{new data})}\end{aligned}$$

The variables A and B need not be observed: they can be a latent representation.

1. A causal model contains more information than a probabilistic model.



2. A joint probability on (X, Y) is consistent with both $X \rightarrow Y$ and $X \leftarrow Y$:

$$p(x, y) = p(x)p(y|x) = p(y)p(x|y).$$

To infer the direction of causality, we can rely on:

- Interventions: change x and check if y changes;
- The “independence” of cause p_X and mechanism $p_{Y|X}$ – for a notion of independence to be defined;
- The independence of the noises in the **structural causal model** (SCM):

$$\begin{aligned} X &= N_X \\ Y &= f(X, N_Y) \\ N_X &\perp\!\!\!\perp N_Y. \end{aligned}$$

(The influence of time is not always clear: in psychology, cause and effect tend to be simultaneous; at the microscopic scale, the laws of physics are invertible.)

3. To model an *intervention* with a SCM, just change the corresponding assignment.

To study *counterfactuals* with an SCM

$$\begin{aligned} X &= N_X \\ Y &= f(X, N_Y), \end{aligned}$$

observe (x, y) , compute N_X and N_Y , then change the value of X (equivalently, N_X) and see how it affects Y . Counterfactual statements are not transitive ($X = 2 \Rightarrow Y = 5$, $Z = 2 \Rightarrow Y = 10$, $X = 2 \not\Rightarrow Z = 10$).

Do not confuse conditional probabilities, interventions (do-calculus) and counterfactuals: even if the interventional probabilities are the same, the counterfactual statements can differ.

4. Causal inference is possible with just two variables, but requires additional assumptions.

$$\begin{array}{ccc} C & \longrightarrow & E \\ \text{cause} & & \text{effect} \end{array}$$

Indeed, every joint distribution (X, Y) can be written $Y = f(X, N_Y)$, $X \perp\!\!\!\perp N_Y$: just set $f(x, n_Y) = F_{Y|X}^{-1}(n_Y)$, $N_Y \sim U(0, 1)$.

One can restrict the class of models, so that they can only be written in one direction (and hope that this direction is the “causal” one).

- Linear models with non-Gaussian additive noise (LiNGAM), $E = \alpha C + N_E$, $N_E \perp\!\!\!\perp C$;
- Non-linear additive noise model (ANM), $Y = f(X) + N_Y$, $N_Y \perp\!\!\!\perp X$;
- Discrete additive noise;
- Post-nonlinear model, $Y = g(f(X) + N_Y)$, $N_Y \perp\!\!\!\perp X$.

One can choose the direction in which P_C and $P_{E|C}$ are “independent”, in some sense.

- For a deterministic relation,

$$Y = f(X), \quad f: [0, 1] \rightarrow [0, 1], \quad f(0) = 0, \quad f(1) = 1,$$

IGCI (information geometric causal inference) independence is defined as $\text{Cor}[\log f', p_X] = 0$ (where $\log f'$ and p_X are seen as probability densities). We then have $\text{Cor}[\log(f^{-1})', p_Y] \geq 0$, with equality if $f = \text{id}$ (the peaks of p_Y are regions where f has small slope and f^{-1} large slope: p_Y contains information on f^{-1}).

- In high dimension, the model $Y = AX + N_X$, $N_X \perp\!\!\!\perp X$, $A \in \mathbf{R}^{e \times d}$ satisfies the *trace condition* if $\tau_e(A \Sigma_{XX} A') = \tau_d(\Sigma_{XX}) \tau_e(AA')$, where $\tau_k(B) = \text{trace}(B)/k$ (it measures the “independence” of A and P_X). If A is invertible and $N_X = 0$, then $\tau(A^{-1} \Sigma_{YY} A^{-1'}) \leq \tau(\Sigma_{YY}) \tau(A^{-1} A^{-1'})$, with equality if all the singular values of A have the same absolute value.
- P_C and $P_{E|C}$ are *algorithmically independent* if the algorithmic mutual information

$$I(P_C : P_{E|C}) := K(P_C) + K(P_{E|C}) - K(P_{C,E})$$

is zero, where K is the Kolmogorov complexity. This implies $K(P_C) + K(P_{E|C}) \leq K(P_E) + K(P_{E|C})$ (it is easier to describe $C \rightarrow E$ than $E \rightarrow C$).

Implementation:

- Regression with subsequent independence tests (RESIT), for additive noise models, estimates $y \sim x$ and $x \sim y$, and tests $x \perp\!\!\!\perp y - \hat{y}$ and $y \perp\!\!\!\perp x - \hat{x}$, with an independence test looking beyond correlation, e.g., HSIC (in R, use `mgcv::gam(y~x)` and `dHsic::dhsic.test`). Alternatively, compute the log-likelihoods

$$L_{X \rightarrow Y} = -\log \text{Var } X - \log \text{Var}(y - \hat{y}).$$

- For IGCI, estimate $C_{X \rightarrow Y} = \int_0^1 \log f'(x) p(x) dx$ as

$$\hat{C}_{X \rightarrow Y} = \frac{1}{N-1} \sum_{j=1}^{N-1} \log \left| \frac{y_{j+1} - y_j}{x_{j+1} - x_j} \right|,$$

where $x_1 < x_2 < \dots < x_N$; it should be smaller in the causal direction; one could also use $H(X)$ instead of $C_{X \rightarrow Y}$

- For the trace method, compute

$$r_{X \rightarrow Y} = \frac{\tau(A_Y \Sigma_{XX} A_Y')}{\tau(A_Y A_Y') \tau(\Sigma_{XX})}$$

it is closer to 1 in the causal direction.

One could train a machine learning model, using datasets (or *dataset features*) as inputs.

5. *Semi-supervised learning* does not work well if $X \rightarrow Y$ (but even in this case, knowing P_X can help build a classifier with lower loss: the loss involves \hat{p}_X).

Covariate shifts can help, if a change in $P_{X,Y}$ is better explained by a change in P_X (with $P_{Y|X}$ unchanged) than by a change in P_Y .

6. SCMs satisfy the *Markov property*, equivalently:

- d -separation implies conditional independence;
- Any X is independent of its non-descendants given its parents;
- $P(X) = \prod P(X_i | \text{Pa}(X_i))$.

Two graphs are in the same Markov equivalence class if they have the same skeleton and the same v -structures (or “immoralities”: two non-directly-linked parents).

The *completed partially directed acyclic graph* (CPDAG) of a Markov equivalence class contains a directed edge $i \rightarrow j$ iff one of the DAGs in the equivalence class has it.

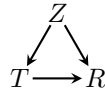
The *Markov blanket* of a node contains its parents, its children, and its children’s parents.

A probability distribution is *faithful* wrt a graph if conditional independence implied d -separation.

A causal effect is *confounded* if

$$P^{\text{do } X=x}(y) \neq P(y | X = x).$$

In the kidney stone example (Z is the size of the stone, large or small, T the treatment, invasive or not, and R , the outcome),



the average causal effect (**ACE**) is

$$P^{\text{do } T=A}(R=1) - P^{\text{do } T=B}(R=1) \\ \neq P(R=1 | T=A) - P(R=1 | T=B)$$

(the signs can even be different: Simpson’s paradox), but we can adjust for Z :

$$P^{\text{do } T=A}(R=1) = \sum_z P(R=1 | T=A, Z=z) P(Z=z).$$

When those equalities hold, Z a *valid adjustment set* of $T \rightarrow R$.

The parents of X form a valid adjustment set for Y (if $Y \notin \text{Pa}(X)$) More generally, Z is a valid adjustment set if it blocks all non-directed paths from X to Y and contains no descendant of any node on a directed path from X to Y .

In a linear Gaussian model, the effect of X on Y can be summarized by a single number,

$$\frac{\partial}{\partial x} E^{\text{do } X=x}[Y].$$

The *propensity score* compresses the influence of several confounders Z into a lower-dimensional variable L .



Pearl’s *do-calculus* is a list of (three) rules saying when we can add or remove a condition, or a “do” action, or replace an action by a condition.

To check *interventional equivalence*, it suffices to consider interventions on single variables, replacing them with noise.

The *unit-level causal effect* is $B_u(t=1) - B_u(t=0)$. The *average causal effect* is

$$CE = \text{Mean}_u B_u(t=1) - B_u(t=0).$$

For a randomized experiment, it can be estimated as

$$\widehat{CE} = \frac{1}{\#U_1} \sum_{u \in U_1} B_u(t=1) - \frac{1}{\#U_0} \sum_{u \in U_0} B_u(t=0).$$

SCMs can be generalized to non-statistical information measures, non-probabilist notions of independence. Let $R : 2^V \rightarrow \mathbf{R}_+$ be a non-decreasing function, measuring the information in a subset $x \subset V$. The conditional information is $R(x, y) - R(y) \geq 0$. The conditional mutual information is

$$I(x : y | z) = R(x, z) + R(y, z) - R(x, y, z) - R(z);$$

it is non-negative if R is modular. We can write $x \perp\!\!\!\perp_R y | z$ when it is zero. For a graph such that $R(x_j, \text{Pa}_j, n_j) = R(\text{Pa}_j, n_j)$ (a node x_j does not contain more information than its parents and its unobserved node n_j) and $R(n_1, \dots, n_d) = \sum R(n_j)$ (the unobserved nodes are independent), d -separation implies conditional independence (wrt R). Examples of such an R include the number of words in a text, or the multi-information,

$$I(x_1, \dots, x_d) = K(x_1, \dots, x_d) - \sum_j K(x_j),$$

where K is the Kolmogorov complexity. The algorithmic independence of conditionals,

$$I(P_{X_1 | \text{Pa}_1}, \dots, P_{X_d | \text{Pa}_d}) = 0$$

is equivalent to

$$K(P_{X_1, \dots, X_d}) = \sum K(P_{X_j | \text{Pa}_j}).$$

7. The DAG is identifiable in the following cases:

- Linear Gaussian with equal error variances;
- Linear non-Gaussian acyclic model (LiNGAM),

$$X_i = \sum_{k \in \text{Pa}_i} \beta_{ik} X_k + N_i;$$

- Nonlinear Gaussian additive noise model

$$X_i = f_i(\text{Pa}_i) + N_i,$$

and its special case, the causal additive model (CAM),

$$X_i = \sum_{k \in \text{Pa}_i} f_{ik}(X_k) + N_i.$$

Known interventions can also help identify the graph.

There are two types of algorithms to infer the structure of an SCM.

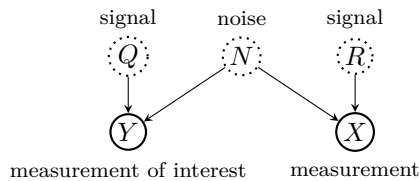
- Some (IC (inductive causation), PC, SGS) test for conditional independence, *i.e.*, d -separation: we can detect v-structures; the CPDAG is identifiable. The conditional independence tests can be based on partial correlation, or be nonlinear: $X \perp\!\!\!\perp Y|Z$ if $X - E[X|Z] \perp\!\!\!\perp Y$ or $Y - E[Y|Z] \perp\!\!\!\perp X$.
- Score-based methods assume that the correct graph gives a better fit to the data and maximize the BIC, or the Bayesian Dirichlet (BD) score (for discrete variables, following a multinomial distribution, with a Dirichlet prior), or the BDe (BD equivalent, accounting for Markov-equivalence). (Greedy equivalence search (GES) uses the BIC, starts with an empty graph, greedily adds edges until a local maximum, then tries to remove some of them, and stops there.)

It is possible to combine score- and independence-based approaches (MaxMin hill climbing). The following can also help infer the structure of the graph:

- Intervention with known targets;
- Intervention with unknown targets (models with a smaller number of intervened variables are more likely);
- Environment changes (we sometimes observe the same SCM, with different parameters).

R software to estimate SCMs include `pcalg`, `bnlearn`, `CompareCausalNetworks`, `InvariantCausalPrediction`, `daggity`.

8. Half-sibling regression was used for exoplanet search.

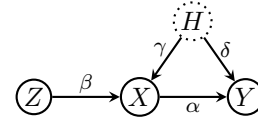


9. Simpson's paradox shows that ignoring hidden variables can lead to incorrect causal conclusions – causal reasoning is very sensitive to model mis-specification.

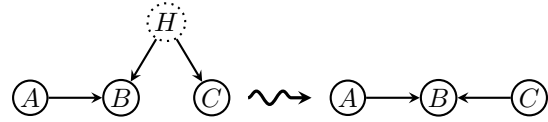
A set of random variables is *causally sufficient* if there are no hidden causes influencing more than one variable.

An *instrumental variable* Z allows us to find the ACE

(average causal effect) α in $Y = \alpha X + \delta H + N_Y$.



If you ignore the hidden variables (and apply the PC algorithm), you can get incorrect causal conclusions.



DAGs are not closed under marginalization; there are many extensions:

- MAG (maximal ancestral graphs);
- PAG (partially ancestral graphs);
- IPG (induced path graph);
- POIPG (completed partially oriented induced path graph);
- ADMG (acyclic directed mixed graph);
- Chain graphs.

The FCI (fast causal inference) algorithm is a generalization of the PC algorithm to PAGs.

Conditional independences are not the only constraints that can help identify a model:

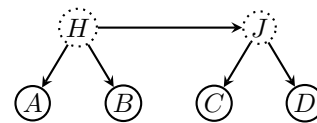
- Verma constraints,

$$\sum_b P(b|a)P(d|a, b, c) = f(c, d)$$

which can detect the presence of a directed edge $A \rightarrow D$;

- Bell's inequality (which ruled out hidden variables in quantum mechanics);
- Elementary inequalities, in information theory, and their generalizations;
- Covariance-based constraints, such as

$$\rho_{AC}\rho_{BD} = \rho_{AD}\rho_{BC} = \rho_{AB}\rho_{CD}$$



- An additive model

$$\begin{aligned} H &= N_H \\ X &= f(H) + N_X \\ Y &= g(H) + N_Y \end{aligned}$$

can be estimated by (nonlinear) dimension reduction;

- One can detect if a path is blocked by a hidden variable with few values.

10. With time series, the DAG is often identifiable, even in presence of instantaneous effects.

Granger causality relies on

$$X_t^k \perp\!\!\!\perp X_{<t}^j | X_{<t}^{-j} \implies X^j \rightarrow X^k$$

(if there are no instantaneous effects). There are non-linear variants of Granger causality. The transfer entropy $TE(X \rightarrow Y) = I(Y_t : X_{<t} | Y_{<t})$ looks tempting, but it fails to measure the strength of an influence.

VAR models can be generalized

$$X_t^i = f_i((Pa_q^i)_{t-q}, \dots, (Pa_0^i)_t) + N_t^j.$$

The **spectral independence criterion** (SIC) is another way of formalizing the independence between cause and mechanism.

$$\begin{aligned} Y_t &= \sum_{k \geq 1} h(k) X_{t-k} \\ C_{XX}(\tau) &= E[X_t X_{t-\tau}] \\ S_{XX} &= \hat{C}_{XX} \text{ (Fourier transform)} \\ S_{YY}(\nu) &= \left| \tilde{h}(\nu) \right|^2 S_{XX}(\nu) \end{aligned}$$

The SIC is

$$\langle S_{XX} \cdot |\tilde{h}(\nu)|^2 \rangle = \langle S_{XX} \rangle \langle |\tilde{h}(\nu)|^2 \rangle,$$

where $\langle f \rangle = \int_{-1/2}^{1/2} f(\nu) d\nu$; one then has

$$\langle S_{YY} \cdot |\tilde{h}(\nu)|^2 \rangle \leq \langle S_{YY} \rangle \langle |\tilde{h}(\nu)|^2 \rangle,$$

with equality if $|\tilde{h}(\nu)|$ is constant.

Machine learning in finance **M.F. Dixon et al. (2020)**

The inconsistent notations make the book needlessly difficult to read.

1. Statistics assumes the data generation process is known, while machine learning does not, but the models are very similar:

- ARIMA and RNN (*i.e.*, NARX) model stationary time series;
- GARCH and GRNN model stationary, heteroskedastic time series;
- CNN generalizes exponential smoothing;
- Auto-encoders generalize PCA.

In supervised learning, a “teacher” provides the right answer; in reinforcement learning (RL), the teacher only provides a “reward”.

The optimal investment problem can be written

$$\begin{aligned} S_{t+1} &= S_t \cdot (1 + \phi_t) && \text{asset price} \\ W_{t+1} &= (1 - u_t)W_t + u_t W_t (1 + \phi_t) && \text{wealth} \\ r_t &= \frac{W_{t+1} - W_t}{W_t} = u_t \phi_t && \text{portfolio returns} \\ R_t &= r_t - \lambda \text{Var}[r_t] && \text{reward.} \end{aligned}$$

The policy u_t maximizing $E \sum R_t$ is

$$u_t^* = \frac{E[\phi_t]}{2\lambda \text{Var}[\phi_t]}$$

Other examples include:

- Algorithmic trading, where the state is the 5-day returns (momentum of all 500 stocks in the S&P 500) and the action whether to buy the S&P500 for 5 days or not;
- Execution: the input is the limit order book (LOB), the output, the next mid-price;
- Mortgage: the output is the mortgage state, the input the previous state and explanatory variables.

2. Neural networks are well-suited to point-wise estimation – we also want to estimate uncertainty.

For frequentists, the source of uncertainty is that we only have a sample, not the whole population (θ is fixed, and $\hat{\theta}$ is a random variable, which depends on the sample).

$$\begin{aligned} \text{MSE}(\hat{\theta}) &:= E \left\| \hat{\theta} - \theta \right\|_2^2 \\ \text{bias}(\hat{\theta}, \theta) &:= E[\hat{\theta} - \theta] = E[\hat{\theta}] - \theta \\ \text{MSE}(\hat{\theta}, \theta) &= \text{tr Var}[\hat{\theta}] + \left\| \text{bias}(\hat{\theta}, \theta) \right\|_2^2 \end{aligned}$$

- If n is large and p small, the frequentist and **Bayesian approaches** are (asymptotically) equivalent.
- If p is large or the likelihood intractable, the Bayesian approach is easier.
- If n is small, prefer the Bayesian approach: frequentist results are often only asymptotic.

For model selection, frequentist tests and AIC assume the models are nested.

If the priors are equal, the Bayes factor is equal to the posterior odds.

$$\begin{aligned} B &= \frac{P(\text{data}|\text{model}_1)}{P(\text{data}|\text{model}_2)} \\ P(\text{data}|\text{model}) &= \int_{\theta} p(x|M, \theta) p(\theta|M) d\theta \\ \frac{P(\text{model}_1|\text{data})}{P(\text{model}_2|\text{data})} &= \frac{P(\text{model}_1) P(\text{data}|\text{model}_1)}{P(\text{model}_2) P(\text{data}|\text{model}_2)} \end{aligned}$$

The Bayesian approach prevents overfitting: if the model is too complex, $P(\text{data}|\text{model})$ is low.

Bayesian model averaging uses $P(\text{model}|\text{data})$ as weights.

Logistic regression is a *discriminative model* estimating $P(y|x)$. The naive Bayes model is the corresponding *generative model*:

$$P(x, y) = p(y) \prod p(x_i | y).$$

3. Bayesian linear regression is a **Gaussian process**:

$$\begin{aligned} \theta_0, \theta_1 &\sim N(0, 1) \\ y &= \theta_0 + \theta_1 x \\ E[y_i y_j | x_i, x_j] &= 1 + x_i x_j \end{aligned}$$

An $\text{AR}(p)$ model is a discrete equivalent of GP regression with a Matern kernel.

Gaussian process regression is available in `sklearn`, `GpyTorch` or `Stan`.

In finance, GP regression can be used for option pricing (train on a grid):

parameters \mapsto price, greeks, precision.

For large amounts of data, use **structured kernel interpolation** (SKI):

- Choose a subset of points U ;
- Use local cubic interpolation: $K_{X,U} \approx W_X K_{U,U}$, with W_X sparse;
- Note that $K_{X,Z} \approx W_X K_{U,U} W_Z'$.

If the inducing points U form a grid and K is an RBF kernel, $K_{U,U}$ is a Kronecker product of Toeplitz matrices: $K_{U,U} = T_1 \otimes \dots \otimes T_p$. Use conjugate gradient (CG) to solve $K^{-1}y$, and compute $\log \det K$ with an approximate diagonalization (Toeplitz matrices are almost circulant, and circulant matrices are diagonalizable with a Fourier transform); rescale to $[0, 1]^n$ for numeric stability. For extrapolation, combine squared exponential (SE) and linear kernels.

The GP also provides analytic derivatives (Greeks): $\partial_{x^*} \mathbb{E}[f|x, y, x^*] = \partial_{x^*} \mu_{x^*} + \partial_{x^*} K_{x^*,x} (K_{x,x} + \sigma^2 I)^{-1} y$

Gaussian processes can have a multi-dimensional response:

$$f \sim \text{MGP}(m\mu, K, \Omega)$$

$$\text{vec}[f(x_1), \dots, f(x_n)] \sim N(\text{vec } M, \Sigma \otimes \Omega)$$

4. The VC dimension of a **feed-forward network** is

$$\text{VC dim} \geq W \times L \times \log \frac{W}{L}$$

where L is the number of layers and W their width.

If the payoff of an option, the option price is a convex function of the spot price: to ensure convexity in a neural net, have the weights of all the layers but the first be positive.

Neural networks can be used in option pricing, e.g., to compute the option price V from the spot price S (with constraints: $V \geq 0$, convex, $0 \leq \Delta \leq 1$) or to predict the implied volatility σ from the moneyness $M = S/K$ (with constraints: $\sigma > 0$, convexity) – more constraints may be needed to avoid arbitrage (e.g., option price increasing wrt T , convex wrt K).

Bayesian neural nets add noise to the weights.

5. To interpret a neural net, compute its **sensitivities**, $\partial y / \partial x$, and $\partial^2 y / \partial x_i \partial x_j$ for the interactions

For instance, the linear risk model $r = Bf + \varepsilon$ can be generalized into a nonlinear risk model $r_t = F_t(B_t) + \varepsilon_t$; the sensitivities can be interpreted as (non-linear) factor returns.

7. The *stochastic volatility model with leverage and jumps* (SVLJ) is

$$y_t = \varepsilon_t \cdot e^{x_t/2} + J_t \pi_t$$

$$x_{t+1} = \mu(1 - \phi) + \phi x_t + \sigma \eta_t$$

$$\begin{pmatrix} \eta_t \\ \varepsilon_t \end{pmatrix} \sim N(0, \Sigma)$$

$$\Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

$$\rho < 0 \text{ (leverage)}$$

$$\pi_t \sim N(0, \sigma^2)$$

$$J_t : \text{ jump, with parameter } p$$

8. The generalized RNN (GRNN) accounts for heteroskedasticity by using the Mahalanobis distance in the loss.

Use stationarity and autocorrelation tests to decide which lag to use in a RNN; for stationary time series, a plain RNN is good enough, for non-stationary ones, GRU or LSTM are preferable.

CNNs generalize weighted moving averages; one can also use non-uniform lags, e.g., 2^k (non-sequential model).

Auto-encoders generalize PCA, for instance to model the yield curve (from the changes in yields).

With ReLU activations, the nonlinear factors learned by an auto-encoder to explain the variability of a portfolio can be built from puts and calls – they are investible.

9. The exploration-exploitation dilemma only appears in online RL – *batch-mode RL*, more common in finance, cannot explore.

SARSA (on-policy) or Q-learning (off-policy) can solve the optimal stock execution problem (linear impact model), after discretizing everything (prices, shares left to sell, time, action):

$$S_{t+1} = S_t(1 - \mu a_t) + \sigma Z_t$$

$$a_t : \text{ shares sold}$$

$$\sum a_t = V$$

$$s_t \geq 0$$

$$r_t = a_t S_t = \lambda \text{Var}[S_{t+1} x_{t+1}] \text{ reward}$$

$$a_t S_t : \text{ value of the shares sold}$$

$$X_{t+1} : \text{ shares left to sell}$$

Portfolio trading (multi-period portfolio management) is another example: the state, $s_t = (p_t, b_t, X_t)$, contains the portfolio composition (stock), the bond investment, and the market state (Markov or, if the lag is too large, HMM). The reward is

$\text{reward}_t = \text{portfolio return} - \lambda \cdot \text{Var}[\text{portfolio return}]$.

The max can be replaced with the *mellow max*

$$\text{mm}(\mathbf{x}) = \frac{1}{\omega} \log \left(\frac{1}{n} \sum e^{\omega x_i} \right).$$

10. Reinforcement learning is trickier in finance, because of the higher dimension and the lower signal/noise ratio.

In option pricing, RL can provide replicating portfolios, minimizing the risk-adjusted cost of hedging:

$$\begin{aligned} H_T(S_T) & \quad \text{payoff} \\ \Pi_t = u_t S_t + B_t & \quad \text{hedge portfolio} \\ u_T = 0 \\ \Pi_T = B_T = H_T(S_T) \\ u_t S_{t+1} + e^{r\Delta t} B_t = u_{t+1} S_{t+1} + B_{t+1} \\ B_t = e^{-r\Delta t} [B_{t+1} + (u_{t+1} - u_t) S_{t+1}] \end{aligned}$$

$$u_t^*(S_t) = \underset{u}{\text{Argmin}} \text{Var}[\Pi_t | \mathcal{F}_t] = \frac{\text{Cov}(\Pi_{t+1}, \Delta S_t | \mathcal{F}_t)}{\text{Var}(\Delta S_t | \mathcal{F}_t)}$$

In higher dimensions, prefer probabilistic methods to Q-learning.

$$\begin{aligned} x_t & \quad \text{portfolio values} \\ b_t & \quad \text{cash} \\ v_t = \mathbf{1}' x_t + b_t & \quad \text{portfolio value} \\ x_t^+ = x_t + u_t & \quad \text{investment decision} \\ \mathbf{1}' u_t + b_t^+ - b_t = 0 & \quad \text{self-financing, } v_t = v_t^+ \\ p_{it} & \quad \text{asset prices} \\ r_t & \quad \text{asset returns} \\ x_{t+1} = (\mathbf{1} + r_t) \odot x_t^+ & \\ r_t - r_f \mathbf{1} = W z_t + M' u_t + \varepsilon_t & \quad \text{asset return model} \\ z_t & \quad \text{predictors} \\ M' u_t & \quad \text{market impact} \\ z_{t+1} = (I - \Phi) \odot z_t + \varepsilon_t^z & \quad \text{OU process} \\ y = \begin{pmatrix} x \\ z \end{pmatrix} & \quad \text{state} \\ R^0 = (r_t - r_f \mathbf{1})'(x_t + u_t) & \quad \text{wealth gain} \\ R = R^0 - \lambda \text{Var}[R^0] - \text{impact} - \text{fee} & \quad \text{reward} \end{aligned}$$

The Bellman equation can be written

$$\begin{aligned} V^*(y) &= \underset{a}{\text{Max}} R(y, a) + \gamma \text{E}[V^*(y')] \\ &= \underset{\pi}{\text{Max}} \underset{a \sim \pi}{\text{E}} [R(y, a) + \gamma \text{E}[V^*(y')]]. \end{aligned}$$

One can then add a penalty

$$\frac{1}{\beta} \log \frac{\pi(a|y)}{\pi_0(a|y)}$$

for each time step

$$\frac{1}{\beta} \sum_{t' \geq t} \gamma^{t'-t} \log \frac{\pi(a_{t'}|y_{t'})}{\pi_0(a_{t'}|y_{t'})}.$$

G-learning is Q-learning with a stochastic policy and an entropy penalty, for noisy environments (the G-function is the Q-function with an entropy penalty).

F-learning is the analogue for the the state value function V . *Soft-Q-learning* is G-learning with a uniform prior π_0 .

RL also applies to wealth management,

$$W_{t+1} = (W_t - c_t)[(1 - \alpha_t)R + \alpha_t R_f]$$

where the policy is $(c_t, \alpha_t)_{0 \leq t < T}$ and the reward CRRA utility.

For instance, for a defined-contribution retirement plan,

$$\begin{aligned} x_t &: \text{asset values (including cash)} \\ u_t &: \text{changes} \\ r_t &: \text{returns} \\ \bar{r}_t &: \text{average returns} \\ \Sigma_t & \\ c_t &: \text{cash inflow} \\ -c_t &: \text{consumption} \end{aligned}$$

the reward is

$$-c_t - \lambda \text{E}[(\text{target} - (1 + r_t)(x_t + u_t))_+] - u_t' \Omega u_t,$$

where the second term is a penalty for missing a target portfolio value (replace it with a quadratic penalty for tractability), and the last term is a regularizer.

11. *Behavioural cloning* does not generalize well: the model does not understand the dynamics of the system. Instead, learn the reward function.

In **inverse reinforcement learning** (IRL), the data is (s, a, s') , not (s, a, r, s') , and the goal is to find both a reward function r and a policy π . The reward function is not well-specified: the optimal policy is unchanged if the reward is transformed as

$$\tilde{r}(a, a, s') = r(s, a, s') + \gamma \Phi(s') - \Phi(s).$$

GAIL (generative adversarial imitation learning) calibrates the reward function so that the solution of the RL problem looks like the demonstration.

We can also learn from failed trajectories (labeled as such) – they are actions to avoid. IRLF (IRL from failure) looks for a policy leading to trajectories close to the successful ones and far from the failed ones.

T-REX (Trajectory-ranked Reward EXtrapolation) learns preferences: not the absolute values $r(s, a)$, but just whether $r(s, a) > r(s', a')$.

Financial applications Infer the reward function of a single agent: - high-frequency trading strategy identification (e.g., for fraud detection by regulators): cluster the strategies using the learned reward functions, rather than arbitrary features, which could be uninformative - reward function of a risk-averse option trader: assuming the trader uses the QLBS model, there is only one parameter to find, the risk aversion - reward function of a portfolio investor (...) Infer the reward function of the "average" investor: - Investor

sentiment State: market data Actions: sentiment (discretized into 3 levels), from news Reward: to estimate – it will yield “useful” features; dimension reduction of market data - Inflow/outflow for a single stock Idem, but the actions are the inflow/outflow for a single stock

Machine learning for factor investing **G. Coqueret and T. Guida (2020)**

The authors provide data: 100 characteristics, uniformized, for 1000 stocks (in the US), monthly, over 20 years.

4. Fama-MacBeth regressions proceed in two steps:

- For each stock, regress the returns against the factor returns, yielding the exposure β_{ik} of stock i to factor k ;
- For each date, regress the stock returns against the exposures, yielding the premium γ_{tk} of factor k at time t .

Note the difference between *characteristics* (e.g., E/P) and *exposures* (e.g., β to the “high E/P minus low E/P ” portfolio).

The factor portfolios are often cryptically called SMB (size), HML (value), WML (momentum), RMW (profitability), CMS (investment), BAB (low risk).

5. The features are autocorrelated but the label (the variable to predict) is not: try to remove the autocorrelation.

Removing noisy observations (not only outliers, but also 60% of the values in the center of the distribution) may help.

10. The **BART** package provides Bayesian additive regression trees (regression trees with a prior which, after Monte Carlo sampling, can be used as ensemble).

12. Combine several models with ensembles, forecast combinations (find weights w , nonnegative and summing to 1, to combine forecasts from several models – the nonnegative constraint has a sparsifying effect) and stacked ensembles.

Use decision trees to forecast the absolute error of a model from macro-economic variables; then, use it as a regime indicator.

13. In portfolio optimization, try to add a diversifying constraint $w'w \leq \delta$ (or a penalty) on the Herfindahl index.

Try to forecast returns scaled by volatility, r/e^σ , instead of returns.

14. To interpret models, check the **iml**, **lime**, **breakDown** packages.

15. To infer a causal graph from data, check the **pcalg**, **CAM** or **baycn** package.

The **CausalImpact** and **bsts** packages fit structural time series models, to detect or test for the presence of regime changes.

The **ReinforcementLearning** package provides Q -

learning on SARS data (state, action, reward, next state), for discrete state and action spaces.

Towards a definition of disentangled representations **I. Higgins et al. (2018)**

A latent representation is disentangled wrt a symmetry group of the set of world states, decomposed as a direct product $G = G_1 \times \dots \times G_n$, if:

- The action of G on the world states extends to an action on the latent representations, making the map

$$\text{World states} \mapsto \text{Latent representations}$$

equivariant;

- The space of latent representations decomposes as $V = V_1 \times \dots \times V_n$, such that each G_i acts on V_i and leaves the other V_j ’s invariant.

Note that:

- The decomposition of G is not unique (e.g., for translations in the plane, any basis gives a decomposition $\mathbf{R}^2 \approx \mathbf{R} \times \mathbf{R}$);
- There may be no non-trivial decompositions of G (e.g., rotations, $\text{SO}(3)$).

If V is a vector space and the action linear, we can further decompose the latent space into irreducible representations.

Disentangling by subspace diffusion **D. Pfau et al. (2020)**

Let M be an orientable Riemannian manifold. The parallel transport along a loop $\gamma : [0, 1] \rightarrow M$, $\gamma(0) = \gamma(1) = x$ at a point $x \in m$, defines a linear map $H_\gamma : T_x M \rightarrow T_x M$ – the *holonomy* of γ . The **holonomy group** at x , $\text{Hol}_x(M)$, is the set of all such maps. If $M = M_1 \times \dots \times M_n$, then $T_x M = \bigoplus T_x M_i$ (orthogonal decomposition) and the action of each $\text{Hol}_x(M)$ leaves each $T_x M_i$ invariant. If M is simply connected and geodesically complete, the converse is true (*deRham decomposition theorem*): if a proper subspace $U \subsetneq T_x M$ is invariant under $\text{Hol}_x M$ for some x , then $M = M_1 \times M_2$, $U = T_x M_1$, $U^\perp = T_x M_2$.

The holonomy group requires all loops: instead, we can get the same information from random walks, and the corresponding diffusion

$$\frac{\partial p(x, t)}{\partial t} = \Delta^0 p(x, t),$$

where Δ^0 is the Laplace-Beltrami operator on functions on M . The Laplacian can be generalized to a Laplacian Δ^1 on TM , and a “second-order connection Laplacian” Δ^2 on $T^*M \otimes TM$. If $M = M_1 \times \dots \times M_n$, then the projections $T_x M \rightarrow T_x M_i$ are in the kernel of Δ^2 (eigenfunctions for $\lambda = 0$).

The Geomancer (genmetric manifold component estimator) algorithm estimates Δ^2 from points in M :

- Build the symmetric nearest neighbour graph;

- For each i , perform PCA on the $x_j - x_i$, $j \in N(i)$, to get local coordinates U_i and local tangent vectors v_j : $x_j - x_i \approx U_i v_j$;
- Compute the graph Laplacian

$$\Delta^0 f = \left(\sum_{j \in N(i)} f_i - f_j \right)_i$$

$$\Delta^1 v = \left(\sum_{j \in N(i)} v_i - Q'_{ij} v_j \right)_i$$

$$\Delta^2 \Sigma = \left(\sum_{j \in N(i)} \Sigma_i - Q'_{ij} \Sigma_j Q_{ij} \right)_i;$$

- The parallel transport Q_{ij} induced by the ambient (Euclidean) space can be estimated from the local coordinates $Q_{ij} = U_{ij} V'_{ij}$, where $U'_j U_i = U_{ij} \Sigma_{ij} V'_{ij}$ is the SVD decomposition;
- To avoid spurious eigenfunctions (skew symmetric matrices, and small eigenvalues from Δ^0), consider the action of Δ^2 on symmetric, zero-trace matrices
 - the eigenfunctions are linear combinations of the desired projections.

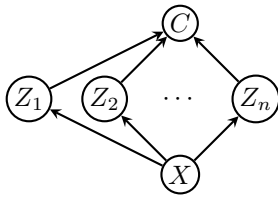
***Robustly disentangled causal mechanisms:
validating deep representations
for interventional robustness***

R. Suter et al.

The VAE penalty can be modified to impose more structure on the latent space: β -VAE (stronger penalty); FactorVAE, β -TCVAE (ICA); DIP-VAE.

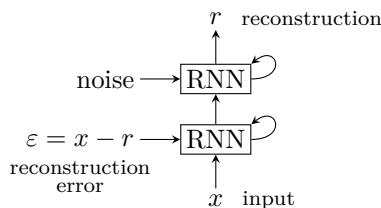
If the “generative factors” G_i are known (orientation, position, shape, colour, etc.), one can try to minimise $MI(G_i, Z_i)$ and minimize $MI(G_i, Z_j)$, $i \neq j$.

We can allow for confounding in the latent factors Z_i by asking that $Z_{\setminus i}$ be a *valid adjustment set* for $Z_i \rightarrow X$.



Towards conceptual compression
K. Gregor et al. (2016)

VAEs output blurry images: *recurrent VAEs* progressively improve the result.



***Reinforcement learning
under moral uncertainty***

E. Ecoffet and J. Lehman (2020)

To ensure robots behave “ethically”, we need a moral theory, but there is no single “correct” one – utilitarianism, deontology and virtue ethics all have their proponents. This *moral uncertainty* can be addressed with multi-agent reinforcement learning and some voting system.

An ethical theory should have an influence over outcomes “proportional” to its credence, regardless of the scale of its rewards (*proportional say*).

In *Nash voting*, each agent (moral theory) has a voting budget and uses part of it, at each step, to vote on the possible actions; the winning theory sees its budget decrease accordingly. Nash voting is unlikely to find compromises.

***Assessing game balance with AlphaZero:
exploring alternative rule sets in chess***

N. Tomašev et al. (2020)

[To plot and compare the entropy of several distributions, show the histogram of the negative log-likelihoods $-\log p$: the average gives the entropy.]

Path signature on Lie groups

D. Lee and R. Ghrist (2020)

The path signature can be generalized to Lie-group-valued time series.

Let G be a Lie group, \mathfrak{g} its Lie algebra, PG the set of paths in G . A path in G , $\gamma : [0, 1] \rightarrow G$, defines a path in \mathfrak{g} , $\gamma' : [0, 1] \rightarrow \mathfrak{g}$; conversely, a path in \mathfrak{g} , $f : [0, 1] \rightarrow \mathfrak{g}$, defines a path in G via the ODE

$$\gamma'_t = f_t(\gamma_t)$$

$$\gamma_0 = g$$

A time series $s(\hat{\gamma}_i)_i$ in G defines a path in G , via the discrete derivative and linear interpolation

$$\hat{\gamma}'_i = \log(\hat{\gamma}_i^{-1} \hat{\gamma}_{i+1}) \in \mathfrak{g}$$

$$\gamma_y = \hat{\gamma}_i \exp[(t - \tau) \hat{\gamma}'_i], \quad t \in [i, i + 1).$$

Note that:

- The logarithm is not always defined: the points should be sufficiently close;
- The exponential is not always surjective (for compact Lie groups, it is);
- The Lie and Riemann exponentials differ (but they coincide for biinvariant metrics; compact Lie groups have such a metric).

The *path signature* of a path $\gamma \in PG$ is a formal power series in \mathfrak{g} , $S(\gamma) \in T((\mathfrak{g}))$ (it can also be defined with

iterated integrals).

$I = (i_1, \dots, i_m)$ multi-index

$N = \dim G$

$1 \leq i_k \leq N$

Δ_m simplex

$\omega_1, \dots, \omega_N$ basis of \mathfrak{g}^*

$$S(\gamma) = \int_{\Delta_m} \omega_{i_1}(\gamma'_{t_1}) \cdots \omega_{i_m}(\gamma'_{t_m}) dt_1 \cdots dt_m \in \mathbf{R}$$

As in Euclidean spaces, the signature characterizes paths up-to tree-like equivalence.

To detect a lead-lag relation between two paths, γ_1 and γ_2 , check the sign of $S^{1,2}(\gamma)$ and $S^{2,1}(\gamma)$, where $\gamma = (\gamma_1, \gamma_2)$.

You may want to transform the time series, e.g.:

- Replacing γ_t with (t, γ_t) , which removes the reparametrization invariance;
- Prepending a path from 0 to γ_0 , which removes translation invariance;
- Replacing γ_t with $(\gamma_t, \gamma_{t-\tau}, \dots, \gamma_{t-m\tau})$.

The signature does not use the Lie algebra structure of \mathfrak{g} . There is a signature-preserving bijection between paths in G and paths in \mathbf{R}^N : we can apply Euclidean data analysis techniques to Lie-group-valued time series.

The (normalized) signature transform is a *universal* and *characteristic* feature map (those properties are equivalent).

- (i) Any continuous bounded function $f : OG \rightarrow \mathbf{R}$ can be approximated by a linear function;

$$f(\cdot) \approx \langle \ell, S(\cdot) \rangle$$

- (ii) The map

$$\begin{cases} \mathcal{M}(PG) & \longrightarrow T(\mathfrak{g}) \\ \mu & \longmapsto E_\mu[S], \end{cases}$$

where $\mathcal{M}(PG)$ is the set of finite Borel measures on PG , is injective.

Fast differentiable sorting and ranking
M. Blondel et al. (2020)

Let $\sigma(\theta)$, $s(\theta)$, $r(\theta)$ denote the arg sort, sort and rank of a vector $\theta \in \mathbf{R}^n$. Arg-sort and rank can be defined as

$$\sigma(\theta) = \underset{\sigma}{\operatorname{Argmax}} \langle \theta_\sigma, \rho \rangle$$

$$r(\theta) = \underset{\pi}{\operatorname{Argmax}} \langle \theta, \rho_\pi \rangle$$

where $\rho = (n, n-1, \dots, 1)$ and θ_σ is θ permuted by σ . The *permutahedron* of a vector $x \in \mathbf{R}^n$ is the convex hull of its permutations.

$$P(w) = \operatorname{Conv}\{w_\sigma, \sigma \in \mathfrak{S}_n\} \subset \mathbf{R}^n$$

Sorting and ranking are solutions of linear programs,

$$s(\theta) = \underset{y \in P(\theta)}{\operatorname{Argmax}} \langle y, \rho \rangle$$

$$t(\theta) = \underset{y \in P(\rho)}{\operatorname{Argmax}} \langle y, -\theta \rangle$$

The optimization problem

$$\underset{\mu \in P(w)}{\operatorname{Argmax}} \langle \mu, z \rangle$$

can be regularized with $Q(\mu) = \frac{1}{2} \|\mu\|^2$, which gives

$$\underset{\mu \in P(w)}{\operatorname{Argmin}} \frac{1}{2} \|\mu - z\|^2$$

(projection onto the permutahedron) or the entropic regularization $E(\mu) = \langle \mu, \log \mu - 1 \rangle$, which gives

$$\log \underset{\mu \in P(e^w)}{\operatorname{Argmax}} \langle z, \mu \rangle - E(\mu) = \log \underset{\mu \in P(e^w)}{\operatorname{Argmin}} \operatorname{KL}(\mu, e^z).$$

To control the regularization strength, multiply Q or E by ε or, equivalently, divide z by ε . Fine-tuning ε is important for some applications (top- k classification), but irrelevant for others. The solution can be computed as $z - v(z_\sigma(z), w)_{\sigma^{-1}(z)}$, where

$$v_Q(s, w) = \underset{v_1 \geq \dots \geq v_n}{\operatorname{Argmin}} \frac{1}{2} \|v - (s - w)\|^2$$

$$e_E(s, w) = \underset{v_1 \geq \dots \geq v_n}{\operatorname{Argmin}} \langle e^{s-v}, 1 \rangle + \langle e^w, v \rangle.$$

These isotonic regressions can be computed efficiently with the “pool adjacent violator” (PAV) algorithm. The Jacobian is easy to compute. The whole algorithm is $O(n \log n)$. Code available (Numpy, Jax, PyTorch, TensorFlow): **fast-soft-sort**.

Applications include rank correlation and trimmed least squares.

***Predicting what you already know helps:
provable self-supervised learning***
J.D. Lee et al. (2020)

Self-supervised pretraining works well if

$$\text{input} \perp\!\!\!\perp \text{pretext task} \mid \text{downstream task}$$

Approximate conditional independence can be quantified with partial correlation.

***PDE-constrained optimization
and the adjoint method***
A.M. Bradley (2010)

The adjoint method computes the gradient, $df(x(p))/dp$, of an implicit function $x(p)$ defined by $g(x, p) = 0$, using the Lagrangian $L(x, p, \lambda) =$

$$f(x) + \lambda' g(x, p).$$

$$\begin{aligned} \frac{df}{dp} &= \frac{d}{dp} L(x(p), p, \lambda) \\ &= \frac{d}{dp} [f(x) + \lambda' g(x(p), p)] \\ &= \frac{df}{dx} \frac{dx}{dp} + \lambda' \left(\frac{\partial g}{\partial x} \frac{dx}{dp} + \frac{\partial g}{\partial p} \right) \\ &= \left(\frac{df}{dx} + \lambda' \frac{\partial g}{\partial x} \right) \frac{dx}{dp} + \lambda' \frac{\partial g}{\partial p} \end{aligned}$$

If dx/dp is difficult to compute, we can choose λ to make the first term disappear.

$$= - \left(\frac{\partial g}{\partial x} \right)^{-1} \frac{df}{dx} \frac{\partial g}{\partial p}$$

We can get the same result directly by differentiating $g(x, p) = 0$, but this is useful for PDE-constrained optimization,

$$\begin{array}{ll} \text{Find} & p, x \\ \text{To minimize} & F(x, p) = \int_0^T f(x(t), p, t) dt \\ \text{Such that} & h(x, \dot{x}, p, t) = 0 \\ & g(x(0), p) = 0 \end{array}$$

where directly computing

$$\frac{dF}{dp} = \int_0^T \left(\frac{\partial f}{\partial x} \frac{dx}{dp} + \frac{\partial f}{\partial p} \right) dt$$

is difficult because of dx/dp . The Lagrangian is

$$L = \int_0^T (f + \lambda h) + \mu' g,$$

and we can choose λ (a function) and μ (a vector) to make the terms in dx/dp disappear: we are left with a differential equation for λ .

Applications of PDE-constrained optimization include:

- Finding the parameters of a PDE given observations;
- Design optimization (e.g., airplane wing).

**Neural CDEs for long time series
via the log-ODE method
J. Morill et al. (2020)**

Neural ODEs are a continuous analogue of ResNets. Neural controlled differential equations are continuous analogues of RNNs.

A *controlled ODE* (CDE) is an equation of the form

$$Z_t = Z_a + \int_a^t f(Z_s) dX_s$$

where $X : [a, b] \rightarrow \mathbf{R}^n$ and f is unknown.

In a neural CDE, X is known (linear interpolation of an irregularly sampled time series), $X + Z$ is a latent (*i.e.*, hidden) variable, and $Y_t = \ell(Z_t)$ is observed; given X and Y , we want to estimate f and ℓ . As for RNNs,

estimation for long time series is challenging. The *log-ODE* method uses

$$Z_t = Z_a + \int_a^t \hat{f}(Z_s) \text{LogSig}_{a,b}^N(X) ds,$$

where $\text{LogSig}_{a,b}^N$ is the depth- N log-signature and \hat{f} extends f to the Lie algebra where $\text{LogSig}_{a,b}^N$ lives (but the implementation directly estimated f and ignores the Lie algebra structure).

**Efficient transformers: a survey
Y. Tay et al (2020)**

Transformers combine vector embedding, positional encoding, self- (and cross-)attention, layer normalization, and residual connections.

The attention matrix is too large for the model to efficiently deal with large sequences, but it can be approximated with fixed patterns (blocks, strides, connected blocks), learnable patterns (e.g., k -means clustering of the tokens), side memory, low-rank factorizations, kernels.

**Tensor programs I:
wide feedforward or recurrent networks
of any architecture are Gaussian processes
G. Yang (2019)**

Random, shallow, infinitely wide neural nets are Gaussian processes. This generalizes to “tensor programs”, involving three types,

- G : vector, asymptotically Gaussian,
- H : vector,
- A : matrix, asymptotically Gaussian,

and three operations,

$$\begin{aligned} \text{MatMul} : & \begin{cases} A \times G & \longrightarrow G \\ (A, x) & \longmapsto Ax \end{cases} \\ \text{LinComb} : & \begin{cases} G^k & \longrightarrow G \\ (x_1, \dots, x_n) & \longmapsto \sum a_i x_i \end{cases} \\ \text{NonLin} : & \begin{cases} G^k & \longrightarrow H \\ (x_1, \dots, x_n) & \longmapsto \phi(x_1, \dots, x_n) \end{cases} \end{aligned}$$

**Tensor programs II:
neural tangent kernel for any architecture
G. Yang (2020)**

“Neural tangent kernel” refers to two results, corresponding to the initialization and the training of a neural net.

- (i) For a shallow network f with random weights, $\Theta(x, y) = \langle \nabla_{\theta} f_{\theta}(x), \nabla_{\theta} f_{\theta}(y) \rangle$ converges to a deterministic kernel as the width of f tends to infinity.
- (ii) During gradient descent, the kernel remains constant and describes the evolution of the neural net:

$$f_t - f_{\text{data}} = e^{-\eta t \Theta} (f_0 - f_{\text{data}}).$$

**Neural tangent kernel:
convergence and generalization in neural nets**
A. Jacot (2018)

Original NTK paper.

**Generative language modeling
for automated theorem proving**
S. Polu and I. Sutskever (2020)

A transformer model (GPT-like, 36 layers) can generate proof for MetaMath:

- Predict a proof step from a goal;
- Pretrain on text (arxiv, Math StackExchange, Github);
- Train on synthetic data (n -digit arithmetic, ring algebra) and real data (set.mm: 38k theorems, 3m steps);
- Also learn a value function

MetaMath is a proof assistant based on string substitution – it is simpler than human-friendly alternatives (Coq, HOL Light, Lean), but its *de Bruijn factor*, the ratio of the length of a formal proof to that of a textbook, is $10 \sim 20$, instead of $1 \sim 3$.

**KarateClub: an API-oriented
open-source Python framework
for unsupervised learning on graphs**
B. Rozemberczki et al. (2020)

Sclearn-like API for community detection, wholegraph embedding, node and graph classification; built on Numpy (dense matrices), Scipy (sparse matrices), gensim (matrix factorization), PyGSP (signal processing on graphs), NetworkX. GraphTool (community detection) and SNAP are more limited.

**Efficiently sampling functions
from Gaussian process posteriors**
J.T. Wilson et al. (2020)

One can sample from a Gaussian process

$$y|X \sim N(0, K_{XX})$$

as follows

$$\begin{aligned}\mu_* &= K_*(K_{XX} + \sigma^2 I)^{-1} y \\ \Sigma_* &= K_{**} - K_{*X}(K_{XX} + \sigma^2 I)^{-1} K_{X*}.\end{aligned}$$

The *Nystrom approximation* approximates K_{XX} using a subset $Z \subset X$ of the data,

$$K \approx K_Z K_{ZZ}^{-1} K'_Z.$$

The *fully independent training conditional* (FITC) method uses an arbitrary set of points (not necessarily among the observations),

$$K \approx K_Z K_{ZZ}^{-1} K'_Z + \text{diag}(K_{XX} - K_{XZ} K_{ZZ}^{-1} K_{ZX}) + \sigma^2 I$$

(it is obtained by integrating those points out

$$p(y|x) = \int_u p(y, u|x, z) = \int_u p(y|u, x, z) p(u|z)$$

and assuming that the variance of the first factor is diagonal). The computations can be simplified using Woodbury's formula.

Besides those *function space approximations*, the *weight space approximation* expresses f as a linear combination of random Fourier features.

$$\begin{aligned}\phi_i(x) &= \cos(\theta_i x + \tau_i) \\ f &= \sum w_i \phi_i \\ w|x, y &\sim N(\mu, \Sigma) \\ \mu &= (\Phi' \Phi + \sigma^2 I)^{-1} \Phi' y \\ \Sigma &= (\Phi' \Phi + \sigma^2 I)^{-1} \sigma^2\end{aligned}$$

If (a, b) is Gaussian, then (*Matheron's rule* – equality in distribution)

$$(a|b = \beta) \stackrel{d}{=} a + \text{Cov}(a, b) \text{Cov}(b, b)^{-1} (\beta - b)$$

can be used to incrementally update a sample from a Gaussian process.

Towards an API for the real numbers
H.J. Boehm (2020)

Recursive real arithmetic replaces real numbers $x \in \mathbf{R}$ with (computable) functions $f_x : \mathbf{R}_+^\times \rightarrow \mathbf{Q}$ where $f_x(e)$ is a rational approximation of x with error bounded by e : $|x - f_x(e)| \leq e$. Equality of recursive reals is undecidable (halting problem), except in some special cases (often, when the computations only involve integers, the 4 operations, log, exp, trigonometric functions and their inverses).

Applications include calculators (Google's Android calculator, for users who do not understand IEEE floating point arithmetic) and testing the accuracy claims of floating point implementations.

Also check F. Johansson's Calcium.

**Traditional and heavy-tailed regularization
in neural network models**
C.H. Martin and M.W. Mahoney (2019)

As the optimization progresses, the spectral density of the weight matrices (distribution of the singular values) changes: Marchenko-Pastur (MP), corresponding to Gaussian weights; MP with outliers, corresponding to Gaussian weights plus a low-rank matrix; power law distribution. Code available.



**A neural network based framework
for financial model calibration**
S. Liv et al. (2020)

Train a neural network to model the mapping from SDE parameters to option price; then, train another one to learn the inverse mapping.

Learning to see through observations

Y.L. Liv et al (2020)

Remove window reflections or fence obstructions from short videos using the motion difference between the obstruction and the background.

Modeling 3D shapes by reinforcement learning

C. Lin et al.

Generate 3D shapes, as humans do in 3D software, in two steps:

- First, approximate the shape with 3D primitives;
- Then, edit the mesh (edge loops rather than vertices) for the details.

Use imitation learning of a heuristic policy as pretraining before actual reinforcement learning.

Stanza: a Python NLP toolkit for many human languages

P. Qi et al. (2020)

If CoreNLP, Flair, Spacy, UDpipe do not cover enough languages (includes Japanese, Chinese, Arabic, but also ancient Greek, old French, etc. – much fewer for NER).

Deep rank-based transposition invariant distances on musical sequences

G. Hadjeres and F. Nielsen

To compute a distance between musical sequences, train a seq2seq autoencoder (or a seq2seq network to transpose motifs) and use a permutation-based distance (rank correlation or Kendall's τ) on the high-level features (only use the ℓ largest features).

QuantGAN:

deep generation of financial time series

M. Wiese et al. (2019)

GAN, with a TCN in the generator, to model volatility, drift and innovations separately and reproduce the stylized facts of time series (fat tails, clustered volatility, leverage $\text{Cor}(r, \sigma) < 0$).

$$(\eta_t)_t \mapsto (\sigma_t, \mu_t, \varepsilon_t)_t \mapsto (\mu_t + \sigma_t \cdot \varepsilon_t)_t$$

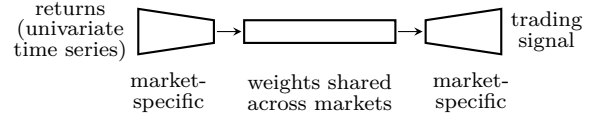
Generative adversarial networks for financial trading strategies fine-tuning and combination

A. Koshiyama et al. (2019)

QuantNet: transferring learning across systematic trading strategies

A. Koshiyama et al. (2020)

Build a stock-level but market-specific (technical analysis) trading strategy using market-specific weights for the first and last layers (linear or LSTM, “encoder” and “decoder”) and shared weights in the middle, with the Sharpe ratio as loss.



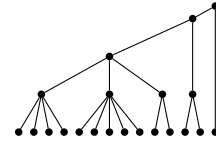
Compact representation of uncertainty in hierarchical clustering

C.S. Greenberg et al. (2020)

Hierarchical clusterings are often built with greedy algorithms, beam search, or sequential Monte Carlo. Consider a (posterior) probability distribution on hierarchies H ,

$$\text{linkage function} \quad E : \mathcal{P}(X) \times \mathcal{P}(X) \longrightarrow \mathbf{R}_+$$

$$\text{energy of a tree} \quad E(H) = \prod_{\substack{X_i, X_j \in H \\ X_i \cup X_j = X}} E(X_i, X_j).$$



The partition function can be computed, exactly, with *dynamic programming*, using

$$Z(X) = \sum_H E(H) = \sum_{\substack{Y \subset X \text{ s.t.} \\ x \in X \\ |X| > 1}} E(X_i, X \setminus X_i) \cdot Z(X_i) \cdot Z(X \setminus X_i).$$

The MAP hierarchical clustering can be computed similarly, using Max instead of \sum (the complexity is “only” $O(2^{2m})$, much less than the number of hierarchies, $(2n - 3)!!$).

Scalable nearest neighbour search for optimal transport

A. Backurs et al. (2020)

The Wasserstein-1 distance on a finite set $X \subset \mathbf{R}^n$ can be used, for instance, to compare documents, seen as distributions on words, after word2vec or glove embedding, but it is expensive to compute. Approximations include:

- Mapping each point in the support of μ to the nearest point in the support of ν ;
- Building a random quad-tree (k-d tree) on X and using

$$W_1(\mu, \nu) \approx \sum_{x \in X} 2^{\ell(x)} |\mu(x) - \nu(x)|$$

where $\ell(x)$ is the level of x in the tree.

Instead, actually compute the optimal transport on the tree,

$$f = \underset{\substack{f \text{ s.t.} \\ \text{pr}_1 f = \mu \\ \text{pr}_2 f = \nu}}{\text{Argmin}} \sum_{x, y \in X} f(x, y) t(x, y)$$

where t is the distance on the tree, and use it with the actual Euclidean distance instead

$$W_1(\mu, \nu) \approx \sum_{x, y \in X} f(x, y) \|x - y\|.$$

***On the limitations
of representing functions on sets***
E. Wagstaff et al. (2019)

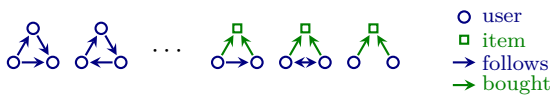
Representing arbitrary permutation-invariant functions with a neural net (deep set) requires a latent representation layer at least the input dimension – dimension reduction would require discontinuous functions.

Evolving normalization-activation layers
H. Liu et al. (2020)

Layer search (as oppsed to architecture search) for a replacement for the BatchNorm-ReLU (or BatchNorm-Swish) combination commonly used as a computation graph made of elementary operations (+, ×, /, max, neg, σ , tanh, exp, log, abs, \cdot^2 , $\sqrt{\cdot}$) and aggregations on some dimensions of the input tensor (batch/channel/instance/group mean/sd) leads to a combination of batch and instance normalization.

***Enhance social recommendation
with adversarial graph convolutional networks***
J. Yu et al. (2020)

Social recommendation systems do not work that well because the relations between users are noisy, heterogeneous and too few. Use an autoencoder to generate *alternate neighbourhoods* (intuitively, informative neighbourhoods, and/or users with similar taste) by applying a GCN (with skip-layer connections) to motif-induced adjacency matrices



to get a user embedding, and then decoding it.

The user embedding and the reconstructed graph can then be used in a discriminator (GCN with attention) to ensure that the alternate neighbourhoods contain the users contributing the most.

***Increasing generality in machine learning
through procedural content generation***
S. Riss et al. (2020)

The interplay between machine learning (ML) and procedural content generation (PCG) goes both ways:

- ML can generate new contents;
- PCG can help train ML systems: data augmentation, domain randomization, curriculum learning, generating new environments (metalearning: POET).

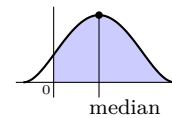
Constructive PCG runs in fixed time, with no search (Perlin noise, L-systems, cellular automata, etc.).

Search-based PCG can use CMA-ES, differential evolution, constraint satisfaction, to search for good contents, sometimes with multiple objective functions (for diversity), which require the computer to play the game for which the contents is being generated.

GANs do not work well: they generate contents that looks right, but does not work (is not playable); they can be combined with search (in the latent space). Reinforcement learning can also be used, to progressively modify random contents into playable contents.

***bayestestR: describing effects and their
uncertainty, existence and significance
within the Bayesian framework***
S. Makowski et al. (2019)

The *probability of direction* (PD) is a Bayesian analogue of a p -value: it is the proportion of the posterior density (of a coefficient in a regression) with the same sign as the median; it varies between 50% (corresponding to $p = 1$) and 100% ($p = 0$). Like the p -value, it does not account for effect size.



In a Bayesian context, the point null hypothesis $H_0 : \beta = 0$ has probability zero and should be replaced by a *region of practical equivalence* (ROPE), e.g., $H_0 : \beta \in [-0.1\sigma, +0.1\sigma]$. The null hypothesis can be rejected if $P_{\text{posterior}}(H_0) \leq \alpha$ (do not use it if the parameters are highly correlated).

The package relies on `rstanarm` and also computes (and plots) MAP estimates, credible intervals and Bayes factors.

***AutoML-Zero: evolving machine learning
algorithms from scratch***
E. Real et al. (2020)

AutoML often limits the search to one aspect of the model, e.g., architecture search using hand-designed block, optimization algorithms, LSTM-like gating mechanisms, data augmentation. A more flexible search space considers three functions, `setup`, `predict`, `learn`; `predict` and `learn` are called in a loop; they are made up of assembler-like operations on scalars (s), vectors (v) and matrices (m), of the same size as the input, with no control structures, e.g.,

$$\begin{aligned} m_1 &= s_2 \times m_2 \\ s_0 &= \text{mean}(v_1) \\ s_3 &= \sin(s_4) \\ s_1 &= \text{runif}(). \end{aligned}$$

The search space is too large for random search, but evolutionary strategies fare better. To speed up the search, use small datasets, and detect equivalent algorithms by hashing their outputs on 10 inputs.

The search rediscovers gradient descent, multiplicative interactions, normalized gradient, weight averaging, dropout, learning rate decay.

This approach can also be used to improve existing algorithms to adapt them to different situations (e.g., little data, fast training, more classes, etc.)

Scaling laws for neural language models
J. Kaplan et al. (2020)

The loss of a transformer model depends on compute time, dataset size, model size.

$$\text{Loss} \propto \text{Compute}^{-0.05} \times \text{Data}^{-0.10} \times \text{Size}^{-0.10}$$

Fawkes: protecting personal privacy against unauthorized deep learning models
S. Shan et al. (2020)

Modify your pictures before distributing them (imperceptibly, in feature space) to prevent their use to train face recognition systems.

FixMatch: simplifying semisupervised learning with consistency and confidence
K. Sohn et al. (2020)

FixMatch performs semi-supervised learning using *pseudolabels* (forecasts of the model, when the model is sufficiently confident) and consistency regularization (similar inputs should give similar outputs).

Stable neural flows
S. Massaroli et al. (2020)

To learn a mapping $u \mapsto v$, *neural ODEs* find a vector field f whose integration gives the desired function.

$$\begin{aligned}\dot{x} &= f(x) \\ x(0) &= u \\ x(1) &= v\end{aligned}$$

More generally, one can add affine maps

$$u \xrightarrow{g} x(0) \mapsto x(1) \xrightarrow{h} v$$

if the dimensions do not match.

$$\begin{aligned}\dot{x} &= f(u, x) \\ x(0) &= g(u) \\ h(x(1)) &= v\end{aligned}$$

The resulting ODE may have stiff dynamics and chaotic behaviour (sensitivity to small perturbations in the input lead to adversarial attacks). Replace $\dot{x} = f(u, x)$ with $\dot{x} = -\partial_x \varepsilon(u, x)$ (or, more generally, a Hamiltonian model).

Dissecting neural ODEs
S. Massaroli et al. (2020)

Vanilla neural ODEs (continuous-depth neural nets) are not universal approximators.

$$\begin{aligned}\frac{dh}{ds} &= f_\theta(s, h) \\ h(0) &= x \\ h(1) &= y\end{aligned}$$

They can be augmented,

$$\begin{aligned}\frac{d}{ds} \begin{pmatrix} h \\ a \end{pmatrix} &= f_\theta(s, h, a) & \frac{d}{ds} \begin{pmatrix} h \\ a \end{pmatrix} &= f_\theta(s, h, a) \\ h(0) &= x & \text{or} & \begin{pmatrix} h(0) \\ a(0) \end{pmatrix} = g(x) \\ a(0) &= 0 & & \\ h(1) &= y & & h(1) = y.\end{aligned}$$

(Higher-order ODEs can also be formulated as augmented neural ODEs.)

Data-controlled neural ODEs incorporate the input x in the vector field,

$$\frac{dh}{ds} = f_\theta(s, h, x).$$

adaptive depth neural ODEs use a data-dependent integration interval

$$\begin{aligned}\frac{dh}{ds} &= f_\theta(s, h) \\ h(0) &= 0 \\ h(s(x)) &= y.\end{aligned}$$

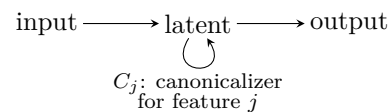
[The implementation uses `torchdiffeq`.]

Representation learning through latent canonicalizations
O. Litany et al. (2020)

Approaches to learn from simulated data include:

- Pretraining on simulated data and fine-tuning on real data;
- Domain adaptation, if there is a lot of unlabeled real data;
- Domain randomization.

To learn a “disentangled” representation (this often just means “isotropic”), train an autoencoder and linear transformations of the latent space which reset one or two parameters of the (generated) input (e.g., colour, background colour, size, rotation, font) to a default value.



$$\begin{aligned}p(x, z) &= p(x|z)p(z) & (\text{generative model}) \\ p(x, z) &= p(x|z) \underbrace{p(z_1) \cdots p(z_n)}_{\text{disentangled}}\end{aligned}$$

Meta-pseudo labels
H. Pham et al. (2020)

A classifier can be trained with the true labels (e.g., binary), smoothed labels, with the labels from a teacher network trained independently (log-probabilities, also available for non-labeled data), or “meta-labels”, still from a teacher network, not trained to produce good forecasts, to help train the student.

Random projections for manifold learning
C. Hegde et al. (2007)

The *Grassberger-Proccaccia algorithm* computes the *intrinsic dimension* of a cloud of points from pairwise distances using the “scale-dependent correlation dimension”

$$C(r) = \text{Mean}_{i \neq j} \mathbf{1}_{\|x_i - x_j\| \leq r}$$

$$\hat{k} = D(r_1, r_2) = \frac{\log C(r_1) - \log C(r_2)}{\log r_1 - \log r_2}$$

where r_1, r_2 are chosen so that C be linear on $[r_1, r_2]$ and $[r_1, r_2]$ be large. It can be computed from a random projection $\mathbf{R}^N \rightarrow \mathbf{R}^M$, with $M \propto K \log N$.

Rethinking batch normalization in transformers
S. Shen et al. (2020)

Since batch statistics, in natural language processing, have much higher variance than in computer vision, layer-norm is often preferred. BatchNorm can be used, but with running statistics instead of batch statistics (“powernorm”).

NeRF: representing scenes as neural radiance fields for view synthesis
M. Mildenhall et al. (2020)

A scene can be represented as a non-convolutional, fully-connected neural net

$$\begin{cases} \mathbf{R}^5 & \rightarrow \mathbf{R}^4 \\ (x, y, z, \theta, \phi) & \mapsto (R, G, B, A) \end{cases}$$

computing the radiance emitted at a point (x, y, z) in direction (θ, ϕ) and turned into an image using volumetric rendering, which is differentiable. It can be learnt from several (100) views of the same scene (processed with COLMAP (SfM)). The density (alpha) only depends on the position, but the colour also depends on the direction, to account for specular effects.

To capture high-frequency details (neural nets are biased towards low-frequency functions, map each coordinate to a high-dimensional space

$$\begin{cases} [-1, 1] & \rightarrow [-1, 1]^{2L} \\ x & \mapsto (\sin 2^0 \pi x, \cos 2^0 \pi x, \dots, \sin 2^{L-1} \pi x, \cos 2^{L-1} \pi x) \end{cases}$$

(this is the Transformer’s positional embedding).

Do not use columetric rendering with fixed points (deterministic quadrature), but with N points sampled at random in evenly-spaced bind (stratified sampling).

Use two networks, a coarse one, with stratified sampling, to estimate the density (alpha channel), as a piecewise constant pdf, and a finer one, for density and colour, using points sampled from this pdf instead of stratified sampling (hierarchical sampling strategy).

On the spectral bias of neural networks
N. Rahaman et al. (2019)

The Fourier transform of ReLU networks can be computed explicitly: lower frequencies are learned first, and they are more robust to random perturbations of the parameters.

Structure from motion
J.L. Schönberger et al. (CVPR 2016)

Structure-from-motion (SfM) reconstructs a 3D scene from an unordered image collection, in several steps:

- Feature extraction: SIFT or learned features;
- Matching: identification of potentially overlapping images;
- Geometric verification and triangulation;
- Adjustments, to avoid error accumulation and down-play outliers.

COLMAP is an open-source implementation, with a series of improvements to increase robustness.

Volatility is rough
T. Gatheral et al. (2014)

Log-volatility looks like a fractional Brownian motion

$$m(q, \Delta) = \langle |\log \sigma_{t+\Delta} - \log \sigma_t|^q \rangle \propto \Delta^{qH}, \quad H \approx 0.1$$

(data from <http://realized.oxford-man.ox.ac.uk>). This explains the power law seen in the skew,

$$\left. \frac{\partial \sigma}{\partial K} \right|_{K=0} \sim \tau^{-(\frac{1}{2}-H)}.$$

Buy rough, sell smooth
P. Glasserman and P. He (2018)

Buy stocks whose implied volatility roughness is high.

The market generator
A. Kondratyev and C. Schwarz

A *restricted Boltzman machine* (RBM) has two layers of binary variables, visible v and hidden h .

$$E(v, h) = -a'v - b'h - v'wh$$

$$p(v, h) \propto \exp -E(v, h)$$

$$\begin{aligned} w &\leftarrow w + \eta[\langle v'h \rangle_{\text{data}} - \langle v'h \rangle_{\text{model}}] \\ a &\leftarrow a + \eta[\langle v \rangle_{\text{data}} - \langle v \rangle_{\text{model}}] \\ b &\leftarrow b + \eta[\langle h \rangle_{\text{data}} - \langle h \rangle_{\text{model}}] \end{aligned}$$

To get an unbiased estimate of $\langle v'h \rangle_{\text{model}}$, one could iteratively sample from $h|v$ and $v|h$, for long enough (10^3 times).

$$p(h_j = 1|v) = \sigma(b_j + w'_{\cdot j}v)$$

$$p(h_i = 1|h) = \sigma(a_i + w'_i h)$$

Instead, k -step *contrastive divergence* (CD) iterates for k steps only:

$$\Delta w_{ij} = \eta[p(h_j = 1|v^{(0)})v_i^{(0)} - p(h_j = 1|v^{(k)})v_i^{(k)}]$$

$$\Delta a_i = \eta[v_i^{(0)} - v_i^{(k)}]$$

$$\Delta b_j = \eta[p(h_j = 1|v^{(0)}) - p(h_j = 1|v^{(k)})]$$

To deal with real (non-binary) data, round number in $[x_{\text{Min}}, x_{\text{Max}}]$ to 16-bit binary numbers.

Experiments on 4 time series of FX returns and 30 hidden units can reproduce the marginal distributions and the correlation structure.

To generate autocorrelated samples, initialize the RBM with the previous values S_{t-1} and sample h and v a small number of times ($\ll 10^3$). To account for non-stationarity (volatility clustering), add a binary indicator for each input time series, indicating if the volatility is high or low (3-month vs long-term), and keep them fixed.

Handling risk-on/risk-off dynamics with correlation regimes and correlation networks
J. Papenbrock and P. Schwendner (2015)

Cluster correlation matrices (estimated at different points in time) to identify regimes: 25 assets (equities, bonds, commodities, FX), daily returns, 1-month window, filtered with average linkage clustering (why not a minimum spanning tree?), k -means, $k = 5$.

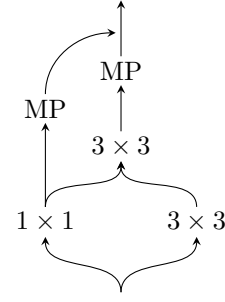
Deep generative models
A. Soleimany (2020)

VAEs can help increase diversity: do not use $N(0, 1)$ to sample in the latent space but something more dispersed (larger variance or fatter tails).

CycleGANs have two generators and two discriminators; they can be applied, not only to images, but also to sounds (to change the voice from one person to another, via spectrograms).

BANANAS: Bayesian optimization with neural architectures for NAS
C. White et al. (2019)

Cell-based search spaces for network architecture search (e.g., from NASBench) assume the network architecture is fixed, with “cells” to be filled in, from a few building blocks; each cell can be encoded by listing the paths in it (there is an exponential number of possible paths, but if the length is limited, it remains reasonable).



For Bayesian optimization, use a neural net (of an ensemble of neural nets) instead of a Gaussian process, to predict (the mean and variance of) the accuracy of an architecture, and Thompson sampling.

Multiplicative interactions and where to find them
S.M. Jayakumar et al. (ICLR 2020)

Try replacing concatenation with multiplication – gating, hypernetworks, dynamic convolutions, attention, etc. Replace linear maps $(x, z) \mapsto W[x; z] + b$ with bilinear ones $(x, z) \mapsto z'Wx + z'U + Vx + b$ (W is then a 3-dimensional tensor).

Big Bird: transformers for longer sequences
M. Zaheer et al. (2020)

The quadratic size of the attention matrix does not allow BERT to scale to large texts. Instead of a dense matrix, use a sparse one, combining:

- Random links: with several layers, random sparse matrices can approximate dense ones;
- A window, *i.e.*, non-zero elements close to the diagonal;
- Bidirectional links between important tokens (e.g., CLS) and all other tokens.

Bootstrap your own latent: a new approach to self-supervised learning
J.B. Grill et al. (2020)

SimCLR performs self-supervised learning by combining augmentations and negative samples. Negative samples are not really needed: use two neural networks, the trained one (“online”) and an exponentially weighted moving average (“target”), the online network predicting the latent representation of the target one under a different augmentation.

Direct feedback alignment scales to modern deep learning tasks and architectures
J. Launay et al. (2020)

Given a neural network

$$\cdots \rightarrow h_{i-1} \xrightarrow{W_i} a_i \xrightarrow{f_i} h_i \xrightarrow{W_{i+1}} a_{i+1} \rightarrow \cdots,$$

back-propagation computes

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_i} &= \frac{\partial \mathcal{L}}{\partial a_{i+1}} \frac{\partial a_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial a_i} \frac{\partial a_i}{\partial W_i} \\ &= \frac{\partial \mathcal{L}}{\partial a_{i+1}} W_{i+1} f'_i(a_i) W_i.\end{aligned}$$

Direct feedback alignment (DFA) replaces the first two factors (since they come from the next layer, they are not biologically plausible, and they prevent parallelization) with

$$\frac{\partial \mathcal{L}}{\partial a_N} B_i$$

where B_i is a fixed random matrix, randomly projecting the global error.

DFA works well for fully-connected architectures (not convolutional ones).

Deep unsupervised learning P. Abbeel et al. (2020)

1. Deep unsupervised learning includes *generative models* which sample from the data manifold, and *self-supervised learning* (e.g., guessing the angle of a rotated image). Those models provide one or more of:

- Probability density;
- Sampling;
- Latent representation.

Applications include (conditional) data generation, anomaly detection, compression (WaveOne is better than JPEG2000), pretraining for text (GPT2, BERT) and vision (CPC, MoCo).

2. *Likelihood-based models* allow us to sample $x \sim p$ and to compute probabilities $p(x)$. For discrete distributions, we could use a *histogram*, but this does not scale (a 28×28 binary image would require $2^{28 \times 28}$ bins) and generalizes poorly. Instead, look for a parametric distribution, estimated using maximum likelihood or, equivalently,

$$\text{KL}(\text{data} \| p_\theta) = \mathbb{E}_{x \sim \text{data}} [-\log p_\theta(x)] - H(\text{data}).$$

If the probability p_θ is modeled by a neural net, we need to ensure that

$$\begin{aligned}\forall x \quad p_\theta(x) &\in [0, 1] \\ \sum_x p_\theta(x) &= 1\end{aligned}$$

This can be done with a *Bayesian net*,

$$\log p_\theta(x) = \sum_i \log p_\theta(x_i | \text{parents}(x_i)),$$

e.g., an **autoregressive model**

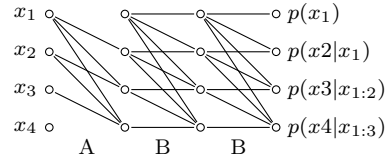
$$\log p_\theta(x) = \sum_i \log p_\theta(x_i | x_{1:i-1})$$

where $p(x_1)$ is given by a histogram and $p(x_i | x_{1:i-1})$ by a neural network with a softmax. *Recurrent networks* allow some parameter sharing – they even work

for MNIST, but try to add skip connections $(i-1, j) \rightarrow (i, j)$ and/or use positional encoding

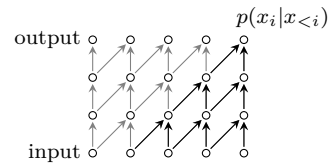
$$(i, j, \text{previous state}) \mapsto (\text{output}, \text{next state}).$$

Masking-based models, such as MADE (masked autoencoder for distribution estimation) take a neural net and remove connections (apply a mask) such that output i only depends on inputs $1 : i-1$.

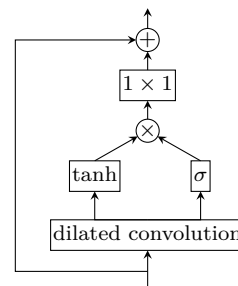
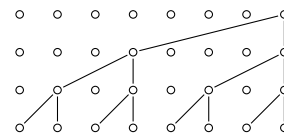


There are two types of layers, A and B, containing only connections from i to j where $i < j$ (A) or $i \leq j$ (B); use at least one layer of type A, anywhere. The mask depends on the ordering; you can use several orderings (masks) on the same underlying network (but not too many).

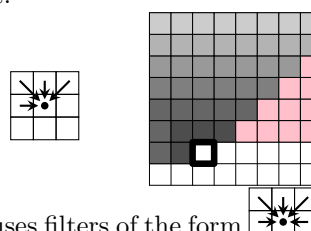
Causal convolutions naturally provide the autoregressive structure, with shared parameters, but they have a limited receptive field.




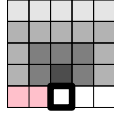
WaveNet uses dilated convolutions, gated residual blocks and skip connections.



A 2-dimensional generalization of WaveNet would have a blind spot.



PixelCNN uses filters of the form  but only above the pixel of interest (“vertical stack”),



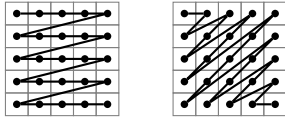
combined with a 1-dimensional network to address the blindspot on the left (“horizontal stack”); the vertical stack is fed to the horizontal stack.

Using a multinomial distribution (softmax) for pixel values is suboptimal: similar pixel values have similar probabilities. PixelCNN++ uses a mixture of logistic distributions (the cdf is a sigmoid), with downsampling and (UNet) skip connections.

Convolutions have a limited receptive field: PixelSNAIL uses CNNs and self-attention,

$$A(q, k, V) = \sum_i \frac{e^{q \cdot k_i}}{\sum_j e^{q \cdot k_j}} v_i$$

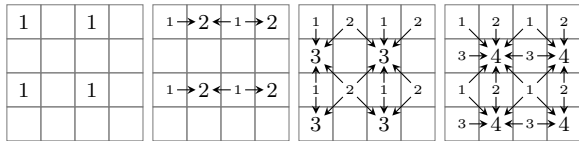
(if the query q is close to the 1-hot vector for coordinate i , the output is close to v_i) with masking; different orders are possible.



Extensions include:

- Conditional PixelCNN;
- Hierarchical AR image models, which progressively increase the resolution;
- Generating a greyscale image first, before adding colour.

AR models are very slow: they generate one pixel at a time. They can be sped up by breaking the AR pattern (*scaling AR video models*).



or by caching the activations (WaveNet, fastPixelCNN).

There is no straightforward latent representation (everything is pixelwise), but the *Fisher score*

$$\dot{\ell}(x, \theta) = \nabla_{\theta} \log p_{\theta}(x)$$

can be used. It also allows interpolation.

3. Flow models allow sampling $x \sim p_{\theta}$, probability computation $p_{\theta}(x)$ and also provide a latent representation. They transform the input into a target distribution, $z \sim p_z$, e.g., $z \sim N(0, 1)$.

$$x \longrightarrow \boxed{} \longrightarrow \boxed{} \longrightarrow \boxed{} \longrightarrow a = f_{\theta}(x)$$

In dimension 1, the distribution of x is given by a change of variable, provided f_{θ} is *invertible*,

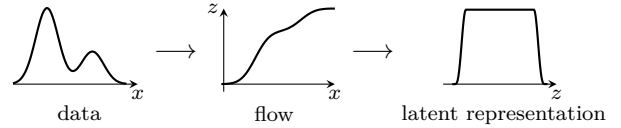
$$p_{\theta}(x) = p_z(f_{\theta}(x)) \left| \frac{\partial f_{\theta}(x)}{\partial x} \right|$$

and θ can be found by maximizing

$$\sum_i \log p_{\theta}(x_i) = \sum_i \log p_z(f_{\theta}(x_i)) + \log \left| \frac{\partial f_{\theta}(x)}{\partial x}(x_i) \right|.$$

Sampling is straightforward: $z \sim p_z$, $x = f_{\theta}^{-1}(z)$.

In dimension 1, with $p_z = U(0, 1)$, the flow is just the cdf of x , and could be estimated as a mixture of Gaussians.

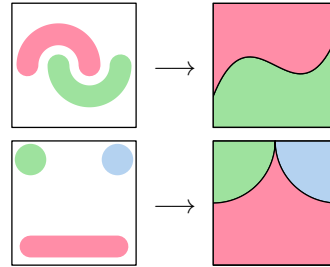


In dimension 2, this becomes

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \begin{pmatrix} z_1 = f_{\theta}(x_1) \\ z_2 = f_{\theta}(x_1, x_2) \end{pmatrix}$$

$$\begin{aligned} \text{Loss}(\theta) = \sum_i \log p_{z_1}(f_{\theta}(x_1)) + \log \left| \frac{dz_1}{dx_1} \right| + \\ \log p_{z_1}(f_{\theta}(x_1, x_2)) + \log \left| \frac{dz_2}{dx_1} \right|. \end{aligned}$$

For instance, $f_{\theta}(x_1)$ could be the cdf of a mixture of 5 Gaussians and $f_{\theta}(x_1, x_2)$ the cdf of a mixture of 5 Gaussians conditional on x_2 .



In dimension n , notice that sampling from a Bayesian net is a flow: use an *autoregressive flow* (AF).

$$\begin{aligned} x_1 &\sim p_{\theta}(x_1) & x_1 &= f_{\theta}^{-1}(z_1) \\ x_2 &\sim p_{\theta}(x_2|x_1) & x_2 &= f_{\theta}^{-1}(z_2; x_1) \\ x_3 &\sim p_{\theta}(x_3|x_1, x_2) & x_3 &= f_{\theta}^{-1}(z_3, x_1, x_2) \end{aligned}$$

$$p_{\theta}(x) = p(f_{\theta}(x)) \left| \det \frac{\partial f_{\theta}(x)}{\partial x} \right|$$

Instead of learning an invertible mapping $x \mapsto z$, one can learn $z \mapsto x$ (inverse AR flow, IAF): training is now sequential (slow), but sampling parallelizable (fast) (ParallelWaveNet, IAF-VAE).

The loss is now

$$\begin{aligned} \mathbb{E}_{x \sim \text{data}} [-\log p_{\theta}(x)] = \\ \mathbb{E}_{x \sim \text{data}} \left[-\log p_z(f_{\theta}(x)) - \log \left| \det \frac{\partial f_{\theta}(x)}{\partial x} \right| \right] \end{aligned}$$

For *affine flows* (multivariate Gaussians)

$$\begin{aligned} z &= f(x) = A^{-1}(x - b) & z &\sim N(0, 1) \\ x &= Az + b & x &\sim N(b, AA') \end{aligned}$$

the log-likelihood is expensive to compute (since $\partial f / \partial x = A^{-1}$, we need $\det A$).

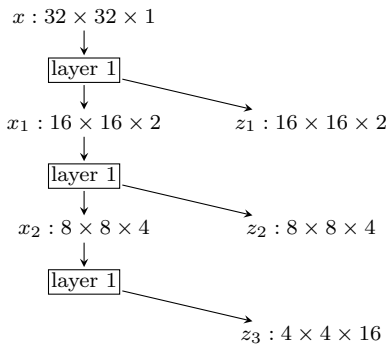
The *elementwise flow* $x \mapsto (f_\theta(x_1), \dots, f_\theta(x_n))$ is not expressive enough, but could be used for some layers.

NICE and RealNVP split the variables in two

$$\begin{aligned} z_{1:d/2} &= x_{1:d/2} \\ z_{d/2:d} &= d_{d/2:d} \cdot s_\theta(x_{1:d/2}) + t_\theta(x_{1:d/2}) \end{aligned}$$

where s_θ, t_θ are arbitrary neural nets.

Split images using a checkerboard pattern or channels.



Flow++ replaces the affine transformations in RealNVP

$$x_i = z_i \cdot a_\theta(\text{parent}(x_i)) + b_\theta(\text{parent}(x_i))$$

with non-linear transformations, e.g., the cdf (or icdf) of Gaussian or logistic mixtures, or piecewise linear or quadratic functions. If the data is discrete $\mathbf{111}$, we do not want a pdf with peaks $\mathbf{\Delta\Delta\Delta}$, but something closer to a piecewise uniform distribution $\mathbf{\perp\perp\perp}$: add $U(-\frac{1}{2}, +\frac{1}{2})$ to the data (*dequantization*). Also check *Glow*, *FFJORD*.

(To check if a latent coordinate k is independent from the others, take two images $x^{(1)}, x^{(2)}$; compute $z^{(1)}, z^{(2)}$; replace $z_k^{(1)}$ with $z_k^{(2)}$; compute the corresponding image x ; check if it looks real, e.g., with a GAN – this could also be part of the loss function.)

4. With AR and flow models, all variables are observed, but they depend on one another. With **latent variable models**, some variables are hidden, but the observed variables are conditionally independent given the latent variables.

$$\begin{aligned} z &\sim p_z \\ x &\sim p_\theta(x|z) \end{aligned}$$

The likelihood is $p_\theta(x) = \sum_z p_z(z) p_\theta(x|z)$ and the model can be trained as

$$\text{Maximize}_\theta \sum_i \log p_\theta(x^{(i)}) = \sum_i \log \sum_z p_z(z) p_\theta(x^{(i)}|z).$$

If z only takes a small number of values, the objective is tractable, e.g., a mixture of 3 Gaussians:

$$\begin{aligned} z &\sim \text{Unif}\{A, B, C\} \\ x &\sim N(\mu_z, \sigma_z^2). \end{aligned}$$

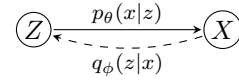
Prior sampling replaces $\mathbb{E}_{z \sim p_z}[\cdot]$ with the average over a sample $z_k^{(i)} \sim p_z$, but most terms are negligible unless, by chance, z_k is close to a latent representation of x – and things get worse in high dimension: use *importance sampling*. A good proposal distribution would be $q(z) = p_\theta(z|x^{(i)})$, but it is difficult to sample from. Instead, use a variational approach and find $q = N(\mu, \sigma^2)$ close to $p_\theta(z|x^{(i)})$ by minimizing

$$\begin{aligned} \text{KL}(q(z) \parallel p_\theta(z|x^{(i)})) &= \mathbb{E}_{z \sim q} \log \frac{q(z)}{p_\theta(z|x^{(i)})} \\ &= \mathbb{E}_{z \sim q} \log \frac{q(z)}{p_\theta(x^{(i)}|z)p(z)/p_\theta(x^{(i)})} \\ &= \mathbb{E}_{z \sim q} [\underbrace{\log q(z)}_{\text{computable}} - \underbrace{\log p_\theta(x^{(i)}|z)}_{\text{does not depend on } q} - \log p(z)] + \log p_\theta(x^{(i)}). \end{aligned}$$

Instead of finding a distribution q minimizing the KL divergence for each i , use *amortized inference* and solve

$$\text{Minimize}_\phi \sum_i \text{KL}(q_\phi(z|x^{(i)}) \parallel p_\theta(z|x^{(i)}))$$

where q_ϕ is a neural net: $q_\phi(z|x) = N(\mu_\phi(x), \sigma_\phi^2(x))$.



Note that p_θ is part of the model while q_ϕ is merely used for inference. This is the *importance weighted autoencoder* (IWAE).

Find θ and ϕ

$$\begin{aligned} \text{To maximize} \quad & \sum_i \log \frac{1}{K} \sum_k \frac{p_x(z_k^{(i)})}{q(z_k^{(i)})} p_\theta(x^{(i)}|z_k^{(i)}) \\ & - \sum_i \text{KL}(q_\phi(x|x^{(i)}) \parallel p_\theta(z|x^{(i)})) \end{aligned}$$

Where $z_k^{(i)} \sim q_\phi(\cdot|x^{(i)})$

The traditional **VAE** uses $K = 1$.

The VAE loss can also be derived from Jensen's inequality:

$$\begin{aligned} \sum_i \log p_\theta(x^{(i)}) &= \sum_i \log \sum_z p(z) p_\theta(x^{(i)}|z) \\ &= \sum_i \log \sum_z \frac{q(z)}{q(z)} p(z) p_\theta(x^{(i)}|z) \\ &= \sum_i \log \mathbb{E}_{z \sim q} \left[\frac{p(z)}{q(z)} p_\theta(x^{(i)}|z) \right] \\ &\geq \sum_i \mathbb{E}_{z \sim q} \log \frac{p(z)}{q(z)} p_\theta(x^{(i)}|z) \end{aligned}$$

Jensen's inequality, $\log \mathbb{E} X \geq \mathbb{E} \log X$, is an equality if X is constant, i.e., if $q(z) \propto p_\theta(z|x^{(i)})$:

$$\sum_o \log p_\theta(x^{(i)}) = \max_q \sum_i \mathbb{E}_{z \sim q} \log \frac{p(z)p_\theta(x^{(i)}|z)}{q(z)}.$$

This is the *variational lower bound* (VLB, or evidence lower bound, ELBO). To train the model, maximize the VLB wrt both q and θ .

The VAE objective can also be derived by computing $\text{KL}(q(z) \parallel p(z|x))$:

$$\log p(x) = \mathbb{E}_{z \sim q} \left[\log \frac{p(z)p(x|z)}{q(z)} \right] + \text{KL}(q(z) \parallel p(z|x)).$$

The optimization wrt q is tricky because the objective is an expectation wrt q . The *likelihood ratio gradient* (score function, REINFORCE)

$$\nabla_\phi \mathbb{E}_{z \sim q_\phi} [f(z)] = \mathbb{E}_{z \sim q_\phi} [\nabla_\phi \log q_\phi(z) \cdot f(z)]$$

is very noisy (try to minimize $\mathbb{E}_{x \sim N(\mu, 1)} \|x - 5\|^2$: the gradient is $\frac{1}{2}(x - \mu) \|x - 5\|^2$ – each step is in a random direction, but they are slightly longer in the right direction). Whenever possible, the *reparametrization trick* (or *pathwise derivative*) works better:

$$\mathbb{E}_{z \sim N(\mu, \sigma^2)} [f(x)] = \mathbb{E}_{\varepsilon \sim N(0, 1)} [f(\mu + \sigma \varepsilon)]$$

The β -VAE forces the latent representation to be even more Gaussian ($\beta \gg 1$):

$$\text{Loss} = \mathbb{E}_{z \sim q_\phi(z|x)} [\log p_\theta(x|z)] - \beta \cdot \text{KL}(q_\phi(z|x) \parallel p(z)).$$

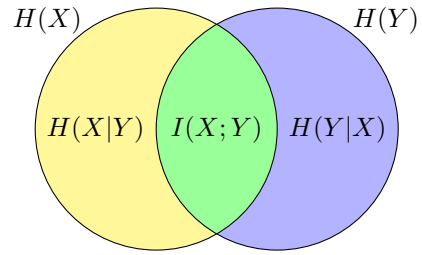
The VQ-VAE provides a better separation between classes by quantizing the latent space. The VQ-VAE-v2 (current SOTA) uses a hierarchical latent space (with progressive up- and down-sampling) and self-attention; contrary to GANs, it does not suffer from mode collapse.

More complex distributions for $p(x|z)$, e.g., PixelCNN, tend to forget the latent code...

Variational approximation has other uses. Flow++ does not use uniform quantization, which results in piecewise constant distributions, $\square\square\square$, but learns it (*variational dequantization*) \triangleleft .

Mutual information can be approximated as

$$\begin{aligned} I(z; x) &= H(z) - H(z|x) \\ &= H(z) - \mathbb{E}_{(z,x) \sim p(z,x)} [-\log p(z|x)] \\ &= H(z) + E[\log q(z|x) - \log q(z|x) + \log p(z|x)] \\ &= H(z) + E[\log q(z|x)] + \text{KL} \\ &\geq H(z) + E[\log q(z|x)] \end{aligned}$$



5,6. Implicit models learn a mapping $z \mapsto x$ without explicit density estimation – just samples. We can no longer compute $\text{KL}(p_{\text{data}} \parallel p_{\text{model}})$: we will use other distances (MMD, JSD, EMD, etc.)

A **generative adversarial network** (GAN) is a min-max game between a generator and a discriminator

$$\min_G \max_D \mathbb{E}_{x \sim \text{data}} \log Dx + \mathbb{E}_{z \sim p_z} \log(1 - DGz).$$

To evaluate the quality of the generated images, the kernel density estimator (aka *Parzen window density estimator*) does not work well in high dimension. The *inception score* uses a pretrained classifier: the class of each generated image should be easy to recognize ($p(y|x)$ has low entropy) and the classes should be diverse ($p(y)$ has high entropy).

$$\begin{aligned} \text{IS} &= \exp[H(y) - H(y|x)] \\ &= \exp \mathbb{E}_{x \sim \text{generated}} \text{KL}(p(y|x) \parallel p(y)) \end{aligned}$$

It is still susceptible to model collapse. The *Frechet inception distance* (FID) compares the mean and covariance of the features (Inception-v3 pool3) of the real and generated images:

$$d^2 = \|m - m_w\|^2 + \text{tr}(C + C_w - 2(CC_w)^{1/2}).$$

Those evaluation methods are not good enough to be used as objectives, though.

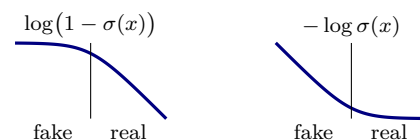
The optimal discriminator is

$$D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{\text{generated}}(x)};$$

the objective is then

$$V(G, D^*) = -\log 4 + \text{JSD}(p_{\text{data}} \parallel p_{\text{generated}}).$$

The Jensen-Shannon divergence is between the KL (no mode collapse, blurry) and the inverse KL (mode collapse) divergences. If the discriminator is too good, it saturates (its gradient is zero – uninformative): for the generator loss (only), change the second term from $\log(1 - DGz)$ to $-\log DGz$: the uninformative gradients are no longer for the fake samples but for the real ones. (It is no longer a zero-sum game: the loss is different for G and D .)



DCGAN (deep convolutional GAN) uses mostly convolutions; it allows for vector arithmetic (pose, smile, glasses, gender). When interpolating using a GAN, do not use a straight line (it would go close to the origin, which is a muddy image), but a half-circle away from the origin.

Improved training of GANs introduced a few ideas:

- Feature matching: for some feature f , add

$$\left\| \mathbb{E}_{x \sim \text{data}} [f(x)] - \mathbb{E}_{z \sim \text{noise}} [f(Gz)] \right\|^2$$

to the loss;

- Minibatch discrimination: feed minibatches, rather than individual images, to the discriminator, to spot mode collapse (of just add $\exp - \|fx_i - fx_j\|_1$ to the loss, or to the input, for some feature f);
- Historical averaging: regularize by adding $\|\theta - \frac{1}{T} \sum_t \theta_i\|^2$ to the loss (cf. Cesaro (usually at text time only), TRPO);
- One-sided label smoothing (only for the real data);
- Virtual batch normalization: batch-norm procudes correlated samples: compute the batch statistics using a virtual batch made of the current observation and a fixed reference batch;
- Semi-supervised learning: have the discriminator predict labels, if available, instead of just real vs fake;
- Inception score.

The *Wasserstein GAN* (WGAN) replaces the JSD with the Wasserstein distance

$$\begin{aligned} W(p_{\text{real}}, p_{\text{gen}}) &= \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(x,y) \sim \gamma} \|x - y\| \\ &= \sum_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_r} [f(X)] - \mathbb{E}_{x \sim p_g} [f(X)]. \end{aligned}$$

To enforce the Lipschitz condition, just clip the weights. There is no classifier: the Lipschitz function is just a “critic”. The gradient is more meaningful, and the model more robust (no need for all those tricks).

WGAN-GP replaces the weight clipping with a gradient penalty $(\|\nabla_x D(x)\|_2 - 1)^2$, evaluated at a random point between a real and a fake sample; there is no batch-norm in the discriminator.

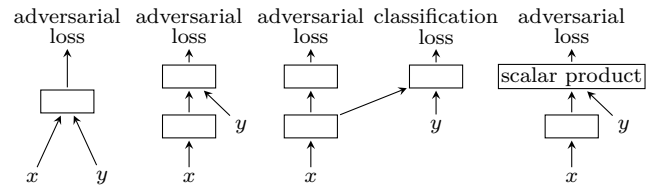
ProGAN progressively increases the resolution.

SNGAN (spectral normalization) uses a *hinge loss*

$$\begin{aligned} L_D &= \mathbb{E}_{x \sim \text{data}} (Dx - 1)_- + \mathbb{E}_{z \sim \text{noise}} (-DGz - 1)_- \\ L_G &= - \mathbb{E}_{z \sim \text{noise}} [DGz], \end{aligned}$$

conditional batchnorm, and enforces the Lipschitz condition using *spectral normalization* of the weight matrices, $W \leftarrow W/\sigma(W)$; the spectral norm (largest singular value) can be approximated with power iteration (one iteration is enough).

The discriminator can use labels in different ways:



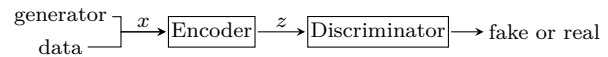
SAGAN uses *self-attention*, spectral normalization (in both discriminator and generator) and hinge loss.

BigGAN scales everythingm adds *orthogonal regularization* $\|W'W - I\|_F^2$, but only for the off-diagonal elements $\|W'W \odot (1 - I)\|_F^2$, use the *truncation trick* (truncate the Gaussian $N(0, I)$ at test time to $[-c, c]$) and “standing statistics” (?) for batch-norm.

StyleGAN uses the noise in a different way: the generator starts with a constant input and progressively upscales it, adding noise at each step; there are two types of noises: the style noise, preprocessed by another network, is incorporated at each step; additional Gaussian noise, different at each step, is also added. As with style transfer, StyleGAN uses (adaptive) *instance normalization*.

StyleGAN2 addresses some artefacts in StyleGAN (e.g., water-droplet-like artefacts, very visible in the activations, but less so in the final images) and can compute the latent representation of an image.

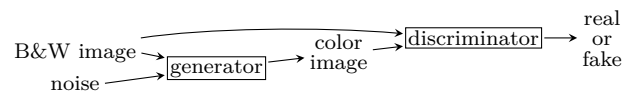
The *variational information bottleneck* (VIB) add another encoder before the discriminator,



to reduce the information it receives, with a constraint on the mutual information $I(X; Z) \leq I_c$ or a variational bound on the mutual information

$$\mathbb{E}_x [\text{KL}(p(z|x) \parallel r(z))] \leq I_c.$$

Pix2Pix is a *conditional GAN* (cGAN) used, e.g., to colour black-and-white images, in which the discriminator takes pairs of images to ensure the output is related to the input;

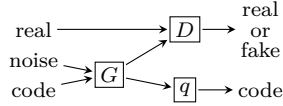


it also uses an autoencoder loss $\text{B\&W} \rightarrow \text{colour} \rightarrow \text{B\&W}$. To reduce the capacity of the discriminator, have it work on patches. Examples include colouring images, turning contours into images, sharpening, turning sketches into photorealistic images (GauGAN), controlling drawing software (learning-to-paint), video (video2video: pose detection to change the person moving or dancing, deep fake adding frames, increasing resolution, etc.)

GANs can also be used for *representation learning*. The *InfoGAN* generator takes as input a code c and noise z to generate an image x ; to ensure that the code is

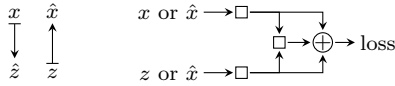
actually used, maximize the mutual information $I(c, x)$ or a variational lower bound.

$$\begin{aligned} I(c; x) &= I(c; G(z, c)) \\ &= H(x) - H(c|G(z, c)) \\ &= H(c) + \mathbb{E}_{x \sim G(z, c)} \mathbb{E}_{c' \sim p(c'|x)} \log p(c'|x) \\ &\geq H(c) + \mathbb{E}_{x \sim G(z, c)} \mathbb{E}_{c' \sim p(c'|x)} \log q(c'|x) \end{aligned}$$



It can also be used for unsupervised category discovery (with a discrete code).

BiGAN (bidirectional GAN) and BigBiGAN learn “generators” in both direction, between image and latent representation; the discriminator learns to distinguish encoder pairs (x, \hat{z}) from generator pairs (\hat{x}, z) .



Energy-based models

$$p(x) = \frac{1}{Z} e^{-E(x)}$$

are estimated by maximizing

$$\mathbb{E}_{x \sim \text{data}} [\log p_\theta(x)] = \mathbb{E}[-E_\theta(x)] - \log Z(\theta).$$

The partition function Z can be replaced by a variational lower bound.

$$\begin{aligned} \log Z &= \log \sum_x e^{-E_\theta(x)} \\ &= \log \sum_x \frac{q(x)}{q(x)} e^{-E_\theta(x)} \\ &= \log \mathbb{E}_{x \sim q} \left[\frac{e^{-E_\theta(x)}}{q(x)} \right] \\ &\geq \max_\phi \mathbb{E}_{x \sim q_\phi} \left[\log \frac{e^{-E_\theta(x)}}{q_\phi(x)} \right] \\ &\geq \max_\phi \mathbb{E}_{x \sim q_\phi} [-E_\theta(x) + H(q_\phi)]. \end{aligned}$$

The objective is then an *entropy-regularized GAN* (the entropy helps prevent mode collapse).

$$\max_\theta \min_\phi \mathbb{E}_{x \sim \text{data}} [-E_\theta(x)] + \mathbb{E}_{x \sim q_\phi} [E_\theta(x)] - H(q_\phi)$$

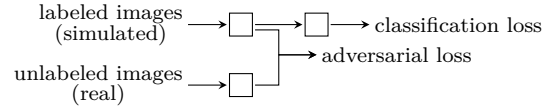
The entropy is usually difficult to compute (except for some models, e.g., PixelCNN).

The WGAN optimizes the dual; *optimal transport* GAN directly optimized the primal, e.g., using *implicit MLE*:

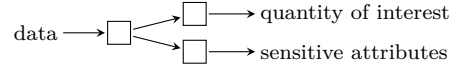
- Generate n samples from p_θ ;
- Pick n real images;

- For each real image, pick the “closest” generated image, form some measure of distance (since each real image is matched, we avoid mode collapse);
- Move θ to make them closer.

Adversarial losses also have applications in *transfer learning*,



fairness,



$$\max_{\theta} \text{Loss}(\theta) \text{ such that } \text{Loss}_{\text{sensitive}}(\theta) \leq K$$

or imitation learning (GAIL)

$$\mathbb{E}_{\pi} [\log D(s, a)] + \mathbb{E}_{\pi_{\text{expert}}} [\log(1 - D(s, a))] - \lambda H(\pi),$$

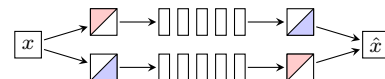
which works better with a variational information bottleneck (VAIL).

7. Self-supervised learning is a form of unsupervised learning which holds part of the data and tasks a neural network to recover it.

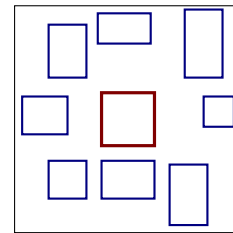
Denoising autoencoders add different types of noise (Gaussian, zero, zero-or-one); they make the output constant in directions orthogonal to the data manifold. They can be stacked.

Impainting removes the central region, a random block, or an element (segmentation mask); use a GAN loss in addition to the reconstruction loss to avoid blurry images.

The *split-brain autoencoder* predicts one view of an image from another (e.g., colour from greyscale, $L \mapsto (a, b)$); it works better with quantized colours (pixel-wise classification problem).



One can try to predict the relative position of two patches (include gaps, jitter the patches; chromatic aberration can be a problem: it points to the center of the image),



or reorder them, as a jigsaw problem, or predict by how much an image was rotated (4 rotations work better than more).

Word embeddings predict the center of a window (CBOW) or the context of a word (skipgram, with negative sampling).

contrastive predictive coding (CPC) looks for an encoder E such that the mutual information $I(Ex_1; Ex_2)$ be high if x_1 and x_2 are patches from the same image, and low otherwise. Applications include sound, iamges. language, reinforcement learning; data augmentation is important.

Even if you have data, unsupervised pretraining followed by supervised fine-tuning outperforms supervised training.

Instance discrimination looks for a latent representation in which augmented versions (cropped, flipped, etc.) of the same image are close and different images are far apart: MoCo (keep a buffer of negative examples), SimCLR, MoCo-v2.

8. Use AR models if you need a density – but sampling is slow.

Use VAEs if you need both a latent representation and samples.

Use GANs if you want good quality samples (sharp images), fast-training, or image-to-image translation.

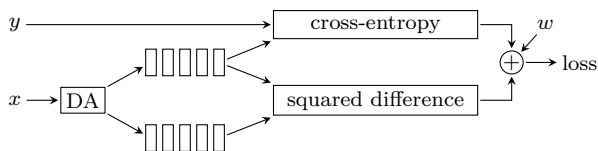
9a. Supervised learning solves

$$\underset{\theta}{\text{Maximize}} \underset{(x,y) \sim \text{data}}{\text{E}} [\log p_{\theta}(y|x)].$$

Semi-supervised learning leverages unlabeled data.

Entropy minimization ensures the classifier is confident on unlabeled data. *Pseudo-labels* are confident predictions, on the unlabeled data, progressively added to the training data.

Label consistency asks that predictions be similar for different augmentations of any given input, e.g., with the π -model



or *temporal ensembling* (the squared difference is with a moving average of past latent representations of the input) or *mean teacher* (instead of keeping a moving average for each input, just keep a moving average of the weights, and compare the latent representations computed with the current and average weights).

An adversarial example $x + r$ can be found as $r = \varepsilon g / \|g\|$, where $g = \nabla_x \log p(y|x; \hat{\theta})$, or

$$r = \underset{\|r\| \leq \varepsilon}{\text{Argmax}} \text{KL}(p(\cdot|x; \hat{\theta}) \| p(\cdot|x+r; \hat{\theta}))$$

(linearize the Kullback-Leibler divergence and use power iteration).

Virtual adversarial training (VAT) makes the forecasts for x and $x + r$, for unlabeled data, close; it performs best.

Unsupervised data augmentation (UDA) (back-translation, image augmentation) improves performance more than random perturbations. *Training*

signal annealing (TSA) helps prevent overfitting on labeled data by masking out high-confidence samples

$$\underset{\theta}{\text{Minimize}} \underset{x,y}{\text{Mean}} - \log p_{\theta}(y|x) \mathbf{1}_{p_{\theta}(y|x) \leq \eta_t}.$$

MixUp creates new, synthetic observations by combining existing ones.

$$\lambda \sim \text{Beta}(\alpha, \alpha)$$

$$\lambda' = \text{Max}(\lambda, \lambda')$$

$$x' = \lambda' x_1 + (1 - \lambda') x_2$$

MixMatch uses MixUp to mix labeled and unlabeled observations, computes the labels by running the classifier on several augmentations of the input, averaging the outputs, and shapening (temperature) the average.

The *noisy student* trains progressively larger models:

- Train a teacher models;
- Train a student model on the original data, and teacher forecasts for unlabeled data, with both input noise (data augmentation) and model noise (dropout, stochastic depth);
- Iterate with a larger student.

9b. Unsupervised distribution alignment tries to find a mapping between two domains, A and B , either as conditional probabilities, $p(a|b)$ and $p(b|a)$, or as maps $A \rightleftharpoons B$, using only unpaired data from A and B . Examples include daylight and night-time images, black-and-white and colour images, segmentation mask and photograph, English and French sentences, image and caption, painting and photograph, etc.

CycleGAN combines *marginal matching*, which approximates the marginals $p(a|b) \approx q_{\theta}(a|b)$, $p(b|a) \approx q_{\theta}(b|a)$ such that

$$q(b) = \underset{a \sim p}{\text{E}} [q(b|a)] \approx p(b)$$

$$q(a) = \underset{b \sim p}{\text{E}} [q(a|b)] \approx p(a)$$

and *cycle consistency*. It can cheat the cycle consistency condition by hiding information (steganography).

DualGAN uses a stochastic mapping $G_{AB} : A \times X \rightarrow B$ instead of $A \rightarrow B$, but the cycle consistency $G)BA(G_{AB}(a, z), z') = a$ makes it ignore z and z' . Instead, the *augmented CycleGAN* learns mappings between $A \times Z$ and $B \times Z$.

Word2vec embeddings for different languages can be aligned with a rotation: approximate marginal matching with adversarial training followed by a refinement of the rotation by solving the exact alignment problem for the top pairs.

Unsupervised machine translation combines word-level alignment, monolingual models (marginal matching), back-translation (cycle consistency).

10. Compression. The *Shannon entropy* is the number of bits needed to transmit information,

$$H(X) = \sum_i p(x_i) \log \frac{1}{p(x_i)}.$$

For any uniquely decodable code C ,

$$\sum_{(s,w) \in C} 2^{-\ell(w)} \leq 1$$

(Kraft-McMillan inequality); conversely, for any lengths satisfying this inequality, there exists a *prefix* code.

The length of a code is at least the entropy, $H(X) \leq \langle \ell \rangle$; there exists a code with average length $\langle \ell \rangle \leq H(X) + 1$. *Huffman coding* (a trie, built bottom up) has length $\ell_i = \lceil -\lg p(x_i) \rceil$; it is optimal.

If we only have an approximation \hat{p} of p , the average length is $\text{KL}(\hat{p}||p) + H(p)$. If the entropy is too high, try conditioning: $H(X|C) \leq H(X)$, e.g., with AR models. The “+1” in $\langle \ell \rangle \leq H(X) + 1$ matters if $H(X)$ is small: use larger chunks. The entropy of the English language is around 1 bit per character.

Arithmetic coding encodes a message as a subinterval of $[0, 1]$: for each new character, split the current interval into subintervals of sizes proportional to the probabilities of each symbol, and pick that corresponding to the new character. To send the final interval, use any number in it.

Asymmetric numeral systems are an alternative to arithmetic coding using integers instead of intervals: partition \mathbf{N} into subsets S_a, S_b , etc., one for each symbol, such that “ S_a contains a proportion $p(a)$ of integers”, and progressively update a state $s \in \mathbf{N}$, starting at $s = 0$, by setting $\text{enc}(s, a)$ to the s th element of S_a .

High-dimensional (discretized) continuous data can be modeled as a mixture of simple distributions, $p(x) = \sum_i p(i)p(x|i)$, using

- Max-mode coding: send $i = \text{Argmax } p(i|x)$ encoded with p , then x encoded with $p(\cdot|i)$; it costs $H(X) + \text{KL}(\text{true}||\text{mixture})$;
- Posterior sampling: use i sampled from $p(\cdot|x)$ instead;
- **Bits-back coding**.

The **Lempel-Ziv** (LZ) compression algorithm looks, in the text seen so far, for the longest match to the string starting at the current position, and outputs its location and length.

11. NLP. The *perplexity* of a language model is the average number of possible next words, *i.e.*, the branching factor – it is between 5 and 10 for free speech, and more constrained, 3 or 4, for translation.

Here are some milestones in the history of language models:

- GloVe, a factorization of the pointwise mutual information matrix,

$$\text{PMI}(x, y) = \log \frac{p(x, y)}{p(x)p(y)},$$

obtained from the co-occurrence matrix;

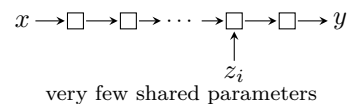
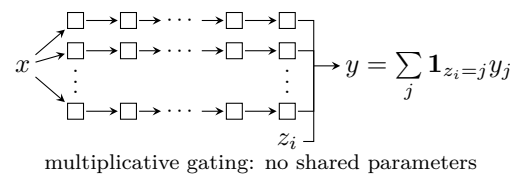
- Word2vec, another vector embedding, predicting a word from its context (CBOW, continuous bag of words) or the opposite (skipgram);

- Byte pair encoding (BPE, usually up to 32,000 tokens);
- n -gram models with smoothing;
- NPLM, an n -gram model, with vector embeddings and a neural net;
- Skip-thought, skipgram with sentences instead of words (contrary to the word embeddings, the sentence embeddings are not model parameters but are computed from the word embeddings);
- ELMO, contextual word representations from RNNs;
- GPT, contextual word representations with a transformer-based generative model;
- BERT (masking), RoBERTa;
- Electra: BERT-like model (masking) with a discriminator to distinguish real from generated sentences;
- T5, which combines all of the above.

Multitask and metalearning

C. Finn (Stanford, 2020)

2. A *multitask learning* network can take as additional input a description of the task,

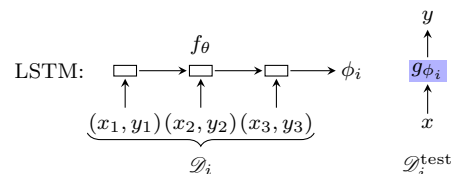


or split the parameters into task-specific and shared (or soft-shared: $\lambda \sum \|\theta_t - \theta_{t'}\|$).

There can be *negative transfer* – independent networks may work better.

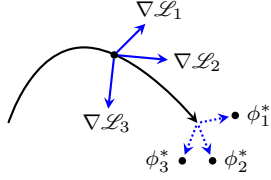
3. There are three types of metalearning algorithms: black-box, optimization-based and non-parametric.

In black-box algorithms, a first network processes the training data and outputs parameters for the second network, which performs the task.



Optimization-based algorithms, e.g., **MAML** (model-agnostic meta-learning) learn parameters θ such that starting the optimization at θ gives the best performance on the new tasks.

$$\text{Minimize}_{\theta} \sum_{\text{task } i} \text{Loss}(\theta - \alpha \nabla_{\theta} \text{Loss}(\theta, \mathcal{D}_i^{\text{train}}), \mathcal{D}_i^{\text{test}})$$



4. MiniImageNet is a metalearning benchmark for one-shot 5-class image classification.

A *siamese network* checks if two images are of the same class – but training a binary classification model for n -way classification may not be optimal. *Matching networks* learn an embedding on which k -NN performs well. If there are several samples per class, a *prototypical network* embeds them all, computes their averages, and uses k -NN on those averages,

$$p(y = k|x) \propto \exp -d(f_\theta(x), c_k).$$

One can also learn the distance (relation networks) or allow for several prototypes per class (infinite mixture of prototypes, IMP).

5. MAML ensembles are an example of Bayesian metalearning algorithm.

6. Since MAML and black-box meta-learning only need gradients, they can be used with policy gradient – but the variance is high, and it is on-policy. Value-based reinforcement learning (off-policy) is not gradient-based: it is difficult to combine it with MAML.

8. Model-based reinforcement learning learns the dynamics (usually, the same across tasks); *model-predictive control* (MPC) replans after each step, because we may end up in a state slightly different from what the model predicted. Learn either in latent space or in image space (video prediction)

$$(\text{observation}_n, \text{action}_n) \mapsto \text{observation}_n.$$

9. In **life-long learning**, the agent is asked to learn a first task, then a second, and so on, instead of all at the same time. Task order can be known, predictable, or random; task boundaries may be known or not; they can change abruptly or progressively shift.

We want to minimize the *regret*.

There can be positive/negative forward/backward transfer: the current task can make the performance on future/previous tasks better/worse.

The *follow-the-leader* algorithm stores all the data seen so far and trains on it. The *catastrophic forgetting* algorithm takes a gradient step on the current observation and forgets it.

To avoid forgetting previous tasks, keep a small amount of data for each task and add a constraint that the loss on the previous tasks does not get worse.

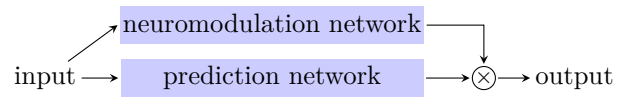
10. *Generative teaching networks* (GTN) learn how to generate data to train on, to help select the network

architecture – once it is chosen, train on actual data, long enough.

Weight normalization suggests to parametrize the weights as $\mathbf{w} = g \cdot \mathbf{v} / \|\mathbf{v}\|$, $g \in \mathbf{R}_+$ and learn \mathbf{v} and g separately.

Learning to RL (LRL – RL² is similar) learns a RNN $(s, t, a_t) \mapsto a_t$; it is amenable to differential Hebbian learning (“neurons that fire together wire together”) and neuromodulated differentiable Hebbian learning (with eligibility traces).

To avoid catastrophic forgetting when “learning to continuously learn”, ANML uses second network to identify the task and zero out the weights irrelevant for it.



POET and Go-Explore use *goal switching* and automatically generate tasks.

11. *Mutual information* is a measure of dependence

$$\begin{aligned} I(x; y) &= \text{KL}(p(x, y) \parallel p(x)p(y)) \\ &= \underbrace{H(p(y))}_{\text{how hard it is to guess } y} - \underbrace{H(p(y|x))}_{\text{how hard it is to guess } y \text{ knowing } x} \end{aligned}$$

For instance, the *empowerment* is

$$I(s_{t+1}; a_t) = H(s_{t+1}) - H(s_{t+1}|a_t).$$

Some variants of Q -learning

$$Q(s, a) \leftarrow s(s, a) + \text{Max}_{a'} Q(s', a')$$

increase exploration (soft Q -learning, SAC)

$$Q(s, a) \leftarrow s(s, a) + \log \int \exp Q(s', a') da'$$

$$Q(s, a) \leftarrow s(s, a) + \underbrace{\mathbb{E}_{s', a'} [Q(s', a') - \log \pi(a'|s')]}_{\text{entropy}}$$

Soft actor critic (SAC) updates the policy as

$$\pi \leftarrow \underset{\pi}{\text{Argmin}} \text{KL}(\pi(\cdot|s) \parallel \exp Q^{\pi^{\text{old}}}(s, \cdot)).$$

To learn without a reward function, train a VAE to represent the state of the world, pick a state at random, and use it as a goal. To get diverse goals, update the VAE using *weighted MLE*, maximizing $\mathbb{E}[w(x) \log p(x)]$ instead of $\mathbb{E}[\log p(x)]$, e.g., with $w(x) = p(x)^\alpha$, $\alpha \in [-1, 0)$, to increase entropy. Learning $\pi(a|s, \text{goal})$ is equivalent to maximizing $I(s; \text{goal})$.

Intrinsic motivation adds an exploration bonus $-\log p_\pi(s)$ to the reward – but check *state marginal matching* (SMM) instead.

Instead of specifying a goal, give the policy a “skill” z (an integer) and have a discriminator try to guess it

from the subsequent state(s). Policy and discriminator cooperate. This maximizes the mutual information $I(z; s)$. (In hierarchical reinforcement learning, once the skills have been learnt, one can learn a policy that switches between them.)

12. To avoid memorization, add a penalty on the information stored in the parameter (*meta-regularization*).

If there is no explicit boundary between tasks, use the performance of the current model to detect changes or, more generally, BOCPD (Bayesian online change-point detection – it is differentiable: you can backprop through it): MOCA (metalearning with changepoint analysis).

If there is no explicit task, learn a representation of the data (unsupervised: BiGAN, DeepCluster), arbitrarily cluster the data (in different ways), and ask metalearning to learn to distinguish those classes.

If two gradients conflict ($g_1 \cdot g_2 < 0$), project each onto the normal hyperplane of the other (PCGrad: projecting conflicting gradients).

Generative deep learning D. Foster (2019)

1. *Naïve Bayes* models are *generative*; they may work with simple tabular data, but will fail with pixels.

2. To classify images (CIFAR-10), one could try dense layers (200, 150, 10, softmax; ReLU), but convolutions (3×3 , 32, 32₁, 32₂, 64₁, 64₁, dense(128), dropout, dense; leaky ReLU, batchnorm after each layer – I indicate the stride with indices) work better.

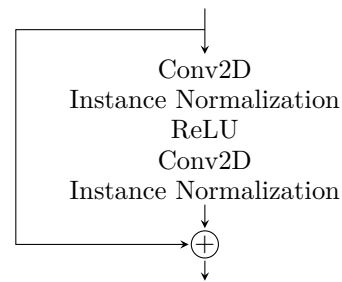
3. To generate images (MNIST), we could try an *autoencoder*, but the distribution in the latent space is not centered, far from Gaussian, and unbalanced. Instead, a *variational autoencoder* (VAE, or β -VAE) adds noise to the latent space and forces the distribution to be close to $N(0, 1)$. Check the 1-dimensional distributions in the latent space: after training, they should look Gaussian. You can interpolate between images, or change some attribute, e.g., by adding $\lambda \cdot [\text{avg}(\text{smile}) - \text{avg}(\neg \text{smile})]$ (CelebA).

4. GANs are notoriously difficult to train: they suffer from mode collapse, oscillating loss, uninformative loss (the loss function of the generator changes with time: if the loss increases with time, this may not be worrying, the quality may be improving) and hyperparameters are tricky to select.

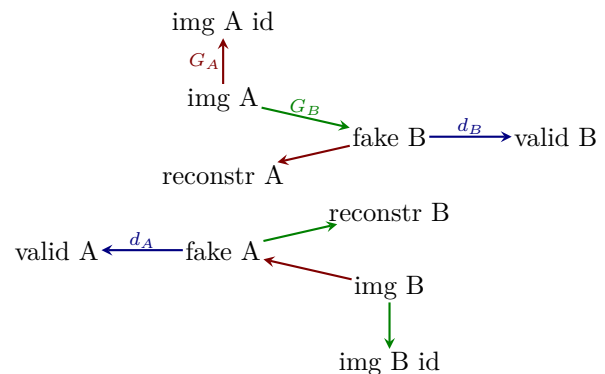
The *Wasserstein GAN* (WGAN) removes the log in the objective, makes the critic 1-Lipschitz with weight clipping, and trains the critic to convergence (e.g., 5 steps for each generator step).

Instead of clipping the weights, the *WGAN-GP* uses a gradient penalty, $(\|\nabla_{\text{input}} \text{critic}\| - 1)^2$, where the gradient is evaluated at random averages between generated and real images, and removes the critic batchnorm. Examples include “Quick, draw!”, CIFAR-10 horses and CelebA.

5. A *CycleGAN* uses two generators and two discriminators, with a UNet or ResNet architecture,



instance normalization instead of batch normalization, and a *PatchGAN* discriminator: it does not output a single number, but an 8×8 matrix indicating if each patch is real or generated (what is consistent across patches is the style, rather than the contents: that is what we want to capture).



Neural style transfer combines three losses: for the contents (MSE of the features), the style (MSE of the GRAM matrices of the features, for several layers), and the total variation (actually $L^{1.25}$ instead of L^1); the model uses a pretrained VGG19 and is fitted with L-BFGS-B [why not gradient descent?].

6. To generate text, try an LSTM RNN, or a GRU, or their bidirectional variants, or stack them, after an embedding layer; train with *teacher forcing*; sample with *temperature* [and try *beam search*] (Maluuba NewsQA, to generate answer-question pairs from text).

7. To generate monophonic music (Bach cello suites), use *music21* to process the MIDI files, and stacked LSTMs, outputting pitch-duration pairs one by one.

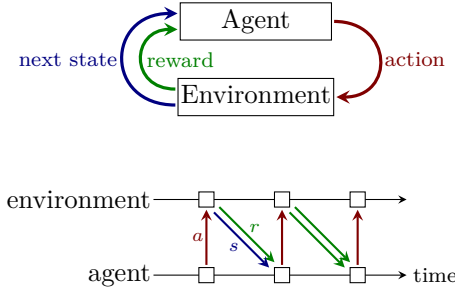
MuseGAN deals with polyphonic music (Bach chorales); the generator has 4 components, whose output change (or not) with each bar (or track); the discriminator uses 3-dimensional convolutions.

8. The *WorldModel* for reinforcement learning (RL) in OpenAI Gym’s CarRacing game has three components:

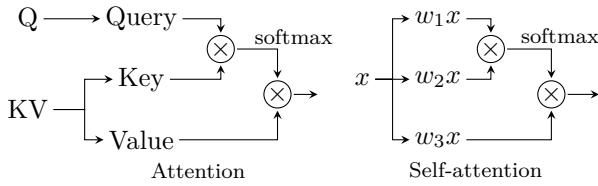
- A VAE to process the image: $\text{img}_t \mapsto h_t \mapsto \text{img}_t$;
- An RNN with a *mixture density network* (MDN-RNN), i.e., an LSTM followed by an MDN (which outputs a mixture of 5 Gaussians in each dimension, i.e., parameters $p_1, \dots, p_5, \mu_1, \dots, \mu_5, \sigma_1, \dots, \sigma_5 \in$

- \mathbf{R}^n), which tries to predict the next state, $(a_{t-1}, h_t) \mapsto \hat{h}_{t+1}$;
- A controller $(a_{t-1}, h_t, \hat{h}_{t+1}) \mapsto (a_t, \hat{r}_{t+1})$, fully connected, trained with *CMA-ES*, in parallel (**estool**).

One can train using the MDN-RNN as a model (*in-dream training*) – to reduce overfitting, add a temperature parameter to control model uncertainty and steer the controller towards safer, better understood actions.



9. Transformers are stacked attention layers (no RNN, no CNN), with *positional encoding* and *multihead attention*. BERT is a bidirectional masked language model; GPT2 is unidirectional; MuseNet is unidirectional and uses a *sparse transformer*.



ProGAN progressively increases the resolution of the image during training.

SAGAN uses self-attention (some parts of the image that are far apart, e.g., the background, should be similar).

BigGAN uses a truncated Gaussian ($z \mid |z| \geq \varepsilon$, instead of $z \sim N(0,1)$), to increase diversity, adds the noise to each layer, uses a shared embedding (?), a hierarchical latent space (?), orthogonal initialization ($W'W = I$), orthogonal regularization ($\|W'W - I\|_F^2$ or $\|W'W \odot (1 - I)\|_F^2$), spectral normalization, hinge loss, EWMA of the weights, etc.

StyleGAN is a style transfer network, based on ProGAN, adding the noise to each layer, with *adaptive instance normalization* to match the mean and variance of each feature with those of the desired style.

Machine learning for asset managers M. López de Prado (2020)

1. Applications of machine learning (ML) in finance include existence (of a phenomenon, of information in data), variable importance, causation, dimension reduction, visualization, latent representation, outlier detection, etc.

2. Use random matrix theory (RMT) to denoise correlation matrices, setting the noise eigenvalues to their average (to preserve the trace).

For clustering, try *detoning* – removing the market component (the matrix is then singular, but it is not an issue).

3. Correlation can be turned into a distance as $d = \sqrt{1 - \rho}$ or $d = \sqrt{1 - |\rho|}$ (it is the Euclidean distance between the standardized vectors). The *variation of information* VI is another distance.

$$H(X) = \mathbb{E}_x[-\log p_x] \quad (\text{entropy})$$

$$= \sum_x -p_x \log p_x$$

$$H(X, Y) = \mathbb{E}_{x,y}[-\log p_{xy}]$$

$$H(X|Y) = H(X, Y) - H(Y)$$

$$\text{KL}(p||q) = \mathbb{E}_{x \sim p} \left[-\log \frac{q_x}{p_x} \right] \quad (\text{KL divergence})$$

$$= \sum_x p_x \log \frac{p_x}{q_x}$$

$$H_c(p||q) = \mathbb{E}_{x \sim p} [-\log q_x] \quad (\text{cross-entropy})$$

$$= H(p) + \text{KL}(p||q)$$

$$I(X; Y) = H(X) - H(X|Y) \quad (\text{mutual information})$$

$$= H(Y) - H(Y|X)$$

$$= \sum_{xy} p_{xy} \log \frac{p_{xy}}{p_x p_y}$$

$$= \text{KL}(p_{xy}||p_x p_y)$$

$$= \mathbb{E}_y [\text{KL}(p(x|y) || p(x))]$$

$$= \mathbb{E}_x [\text{KL}(p(y|x) || p(y))]$$

$$\text{VI}(X, Y) = H(X|Y) + H(Y|X)$$

$$= H(X) + H(Y) - 2I(X; Y)$$

$$= 2(H(X, Y) - H(X) - H(Y))$$

$$= H(X, Y) - I(X; Y)$$

$$\widetilde{\text{VI}}(X, Y) = \frac{\text{VI}(X, Y)}{H(X, Y)}$$

$$= 1 - \frac{I(X; Y)}{H(X, Y)}$$

$$\widetilde{\widetilde{\text{VI}}} = 1 - \frac{I(X; Y)}{\text{Max}\{H(X), H(Y)\}}$$

4. Use *k*-means clustering *on the columns* of the distance matrix $\sqrt{1 - \rho}$ (or $\sqrt{1 - |\rho|}$); choose *k* to maximize the *clustering quality* *q* (optimal number of clusters, ONC)

$$q = \frac{\mathbb{E}[S]}{\text{Var}[S]}$$

$$S_i = \frac{b_i - a_i}{\text{Max}(a_i, b_i)}$$

where a_i (resp. b_i) is the average distance between observation x_i and the elements of its cluster (resp.,

of the nearest cluster). Compute the clustering quality for each cluster; recluster those below average (just once) and keep the new clustering if quality improves.

5. Instead of forecasting labels based on time, tick, volume or price bars,

$$\begin{cases} +1 & \text{if } r > \tau \\ 0 & \text{if } |r| \leq \tau \\ -1 & \text{if } r < -\tau, \end{cases}$$

use a *trippel barrier*

$$\begin{cases} +1 & \text{profit-taking barrier} \\ -1 & \text{stop-loss barrier} \\ 0 & \text{time-out barrier} \end{cases}$$

or *trend scanning*: whether each observation is in an up/down trend or not, by looking at the minimum p -value of linear regressions on $[t, t + T]$, for various T .

For bet sizing:

- Assume the Sharpe ratio is a standard Gaussian and bring it back to $U(-1, 1)$;
- Given n binary predictors, their average under $H_0 : p = \frac{1}{2}$ is known; bring it back to $U(-1, 1)$.

6. To measure feature importance, check:

- p -values;
- Shapley values;
- Accumulated local effects (ALE);
- Mean increase impurity: weighted information gain, from a random forest;
- Mean decrease accuracy: drop in accuracy after shuffling the values of a variable.

Replace the accuracy with:

- The negative log-likelihood of the true labels;
- The probability-weighted accuracy,

$$\frac{\sum_n \mathbf{1}_{\text{correct}} \times (p_n^{\max} - 1/K)}{\sum_n (p_n^{\max} - 1/K)}$$

where K is the number of classes and p_n^{\max} the probability, from the model, of the most likely class.

To deal with multicollinearity, cluster the features.

7. Numeric instability, in portfolio construction, is due to the condition number of the precision matrix V^{-1} ; it is worse when there are highly correlated assets. It is usually dealt with by regularization (Black-Litterman), constraints, or better estimates of the concentration matrix. *Nested clustered optimization* (NCO) denoises (RMT) and clusters (ONC) the covariance matrix, computes optimal portfolios withing each cluster, and then the inter-cluster optimal portfolio.

8. To limit selection bias under multiple testing, estimate the FDR and the FWER from

$$\begin{aligned} \text{FPR} \quad \alpha &= \frac{\text{FP}}{\text{TN} + \text{FP}} \\ \text{FNR} \quad \beta &= \frac{\text{FN}}{\text{TP} + \text{FN}} \\ \theta &= \frac{\text{TP} + \text{FN}}{\text{TN} + \text{FP}} \end{aligned}$$

Approximations of the distribution of

- The sample Sharpe ratio;
- The maximum sample Sharpe ratio after k trials.

help define a *deflated Sharpe ratio*.

Use clustering to estimate the effective number of trials.

9. To limit overfitting, test on synthetic data, using

- Resampling: cross-validation, (sequential) bootstrap and their variants;
- Monte Carlo: parametric, GAN, VAE.

Probabilistic foundations of statistical network analysis H. Crane (2018)

A *random graph* is a graph-valued random variable.

3. We rarely observe the whole graph, but only some *sample* of it: it is not always straightforward to infer properties of the population graph from those of the sample graph (*consistency*), and this depends a lot on how the network was sampled – *vertex sampling* (which tends to generate empty subgraphs, if the population graph is large and sparse), *edge sampling* (a list of edges, e.g., when you intercept communications), *snowball sampling* (pick k nodes at random and consider their radius- r egonets), *hyperedge sampling* (collaboration, e.g., co-author networks) or *path sampling* (traceroute, to study the internet).

4. Common models include:

- Erdos-Renyi: $P(Y_{ij} = 1) = p$;
- Barabasi-Albert (preferential attachment);
- Exponential random graph model (ERGM),

$$P(Y = y) \propto \exp \sum_i \theta_i T_i(y)$$

where the T_i are features (sufficient statistics); ERGMs do not work well with sampled networks;

- Graphons, $P(Y_{ij} = 1) = \phi(U_i, U_j)$, where $U_i \stackrel{\text{iid}}{\sim} U(0, 1)$ and $\phi : [0, 1]^2 \rightarrow [0, 1]$ or, in closed form

$$P(Y = y) = \int \prod_{i \neq j} \phi(u_i, u_j)^{y_{ij}} (1 - \phi(u_i, u_j))^{1 - y_{ij}} du$$

- Sparse graphon, $\phi_n(u, v) = \rho_n^{-1} w(u, v)$, for graphs of size n , where ρ_n^{-1} is the edge density (e.g., $\rho_n = n$ – this allows sparse graphs, which are impossible with graphons, but not power law degree distributions)
- Stochastic block model (SBM) $P(Y_{ij} = 1) = 1 - \exp \theta_i \theta_j w(k(i), k(j))$ degree-corrected SBM $P(Y_{ij} = 1) = 1 - \exp \theta_i \theta_j \log \phi_{k(i), k(j)}(U_i, U_j)$
- Graphex: an exchangeable point process on $[0, \infty)^2$

$$Y(A) = \#\{a \in A : a \in Y\}, \quad A \subset [0, \infty)^\infty,$$

where $(t, t') \in Y$ is a link between t and t' (users are identified by their registration timestamp)

6. Some of those models are *vertex-exchangeable*: the distribution is invariant under permutation of the vertices. More generally, exchangeable random graphs are continuous mixtures of graphons (Aldous-Hoover theorem). This generalizes de Finetti's theorem: if $(X_n)_{n \geq 1}$ is an exchangeable sequence of random variables with values in $\{0, 1\}$ then, there exists $f : [0, 1]^2 \rightarrow [0, 1]$ such that X has the same distribution as Y defined by

$$\begin{aligned} U_n &\stackrel{\text{iid}}{\sim} U(0, 1) & n \geq 0 \\ Y_n &= f(U_0, U_n) & n \geq 1 \end{aligned}$$

i.e., it is a continuous mixture of iid sequences $f_u(U_n) = f(u, U_n)$.

8. *Relatively exchangeable* models only requires that the distribution be invariant under some permutations of the vertices, for instance, generalized SBM

$$\begin{aligned} P(Y_{ij} = 1) &= \phi_{k(i), k(j)}(U_i, U_j) \\ k : \mathbf{N} &\rightarrow \llbracket 1, k \rrbracket \\ \phi_{k_1, k_2} : [0, 1]^2 &\rightarrow [0, 1] \\ U_i &\stackrel{\text{iid}}{\sim} U(0, 1) \end{aligned}$$

where k indicates the class of each node. Relative exchangeability can be relative to another network:

$$P(Y_{ij} = 1) = \phi_{G|_{\{i, j\}}}(U_i, U_j).$$

More generally, consider k -ary relations, for various k 's, $X_1, \dots, X_r, X_j : \mathbf{N}^{a_j} \rightarrow \mathcal{X}_j$, for instance

- $a_1 = 1$, $\mathcal{X} = \llbracket 1, k \rrbracket$, for the SBM;
- $a_1 = 2$, $\mathcal{X} = \{0, 1\}$ for graph exchangeability;
- $a_1 = 1$, $\mathcal{X} = \mathbf{R}^d$ for latent space models.

Under reasonable assumptions, relative exchangeable random graphs are continuous mixtures of

$$P(Y_{ij} = 1) = \phi_{X_1|_{\{i, j\}}, \dots, X_r|_{\{i, j\}}}(U_i, U_j) \quad U_i \stackrel{\text{iid}}{\sim} U(0, 1).$$

9. An edge-exchangeable random graph is an exchangeable sequence of ordered pairs (exchangeable modulo vertex renaming). If \mathcal{P} is the set of vertices, an *interaction propensity process* is a sequence of iid random variables with values in $\mathcal{P} \times \mathcal{P}$,

$$P[X_k = (r, s)] = f_{r, s}.$$

Examples include the vertex component model $f_{ij} = f_i^{\text{out}} f_j^{\text{in}}$ or the (preferential attachment) *Hollywood model* (which does not generate a graph but a sequence of edges). Edge-exchangeable graphs are continuous mixtures of interaction propensity processes.

One can also consider *relative edge exchangeability* $P[X_i = (r, s)] = f_{r, s}^{k(i)}$, $k : \mathbf{N} \rightarrow \llbracket 1, k \rrbracket$. The notion of exchangeability (and the Hollywood model) can be extended to hyperedges (collaboration networks (IMDb, coauthors): the interaction propensity process is a probability distribution on $\text{fin } \mathcal{P}$, the set of finite multisets of elements of \mathcal{P} : $P[X_i = 1] = f_x$, $x \in \text{fin } \mathcal{P}$) and paths.

11. To model dynamic networks, one often makes a simplifying Markov assumption, or even a *projective Markov* assumption ($(G_t)_t$ is Markov on $\{0, 1\}^{N \times N}$, and so are its restrictions). The *temporal ERGM* is

$$P(Y_{t+1} = y' | Y_t = y) \propto \exp[\eta \cdot T(y, y')].$$

A *rewiring process* samples a new rewiring map

$$W : \begin{cases} \{0, 1\}^{N \times N} & \longrightarrow & \{0, 1\}^{N \times N} \\ Y_t & \longmapsto & Y_{t+1} \end{cases}$$

for each t .

A *graph-valued Lévy process* is defined by

- $Y_0 = 0 \in \{0, 1\}^{N \times N}$;
- Stationary increments $Y_{t+s} \triangle Y_t \stackrel{d}{=} Y_s$;
- Independent increments: $Y_{t_0} \triangle Y_{t_1}, Y_{t_1} \triangle Y_{t_2}, \dots, Y_{t_{k-1}} \triangle Y_{t_k}$ independents

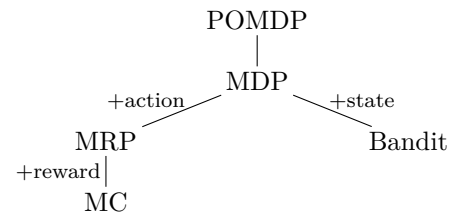
It can be constructed as $Y_0 = 0$, $Y_{t+1} = Y_t \triangle Z_{t+1}$, $Z_{t+1} \sim \mu$, for some distribution μ on $\{0, 1\}^{N \times N}$ (or using a rewiring process).

Deep reinforcement learning hands-on M. Lapan (2020)

1. While the inputs of supervised learning algorithms are pairs (x, y) , where y is the value to predict from x , reinforcement learning uses tuples (s, a, r, s') , where s is the current state, a the action chosen by the agent, r the (immediate) reward and s' the next state, and tries to find the best action for any given state (an action that may not have been chosen by the agent) for the sum of the discounted rewards (and not just the immediate reward).

Challenges of RL include:

- The best action is not known;
- Delayed rewards;
- Non-iid observations;
- Exploitation vs exploration.



2. OpenAI Gym provides RL environments. Some of the environments require the commercial MuJoCo physics simulator (PyBullet is a free alternative, slower with an unpleasant API).

There is no sklearn-like standard Python library for RL algorithms: check OpenAI Baselines (Tensorflow, with a declarative interface), ptan (used by the book), catalyst, etc.

4. The RL algorithm taxonomy distinguishes between:

- **Policy-based** (we directly try to find the best policy, i.e., a map from states to actions – it could be stochastic) vs **value-based** (we try to estimate the

value of each state, *i.e.*, the cumulated reward we would get if we started in a given state and acted optimally – from there, we can find the optimal policy); there are also hybrid algorithms, estimating both policy and value function, each helping improve the other;

- **On-policy** (the training data was generated by the current best estimate of the policy) vs **off-policy** (the training data was generated by some other policy, *e.g.*, an old policy, or a policy modified to explore more).
- **Model-based** (we use the data to estimate the dynamics of the environment) vs **model-free** (we do not) – RL usually refers to searching for an optimal policy for an unknown MDP, only accessible through observations: if the MDP is known, this is an easier **planning** problem.

The **cross-entropy method** (CEM) is the simplest policy-based RL algorithm: it learns to repeat actions that performed well.

- Start with a random policy;
- Act according to this policy;
- Keep the best 30% episodes;
- Train a new policy to replicate what the agent did in those episodes.

This works in simple environments, such as CartPole, but even for FrozenLake, a few adjustments are needed:

- Larger batches (successful episodes are rarer);
- Keep successful episodes in memory for longer;
- Decrease the learning rate; train for longer.

5. For a known MDP, the value of a state $V(s)$, or of a state-action pair $Q(s, a)$, satisfies the **Bellman equation**,

$$V(s) = \max_a E[r + \gamma V(s')] \\ Q(s, a) = E[r + \gamma \max_{a'} Q(s', a')].$$

6. For an unknown MDP, the update

$$Q(s, a) \leftarrow r + \gamma \max_{a'} Q(s', a')$$

is a bit too violent: it completely discards the previous value of $Q(s, a)$... (Tabular) Q-learning uses:

$$\text{target} \leftarrow r + \gamma \max_{a'} Q(s', a') \\ Q(s, a) \leftarrow (1 - \lambda)Q(s, a) + \lambda \text{target}$$

Deep Q-learning (DQN) minimizes

$$\text{Loss} = [Q(s, a) - (r + \gamma \max_{a'} Q(s', a'))]^2.$$

As often with RL, it does not work in practice without a few adjustments:

- Use ε -greedy, with ε decreasing, to explore enough at the beginning;
- Use a replay buffer (fixed-size, or prioritized) to have enough diversity;

- Training is unstable because we are “chasing our tail” ($Q(s, a)$ and $Q(s', a')$ are too close): use a *target network*, and only update it from time to time;
- To ensure the Markov property, keep the previous state(s) in memory.

8. DQN can (should) be improved on:

- N -step DQN (this is no longer really off-policy: we assume that the intermediate action was optimal – but if you only look 2 or 3 steps ahead, it is almost off-policy)

$$Q(s_t, a) = r_t + \gamma \max_a Q(s_{t+1}, a) \\ = r_t + \gamma r_{t+1} + \gamma^2 \max_a Q(s_{t+1}, a) \\ = \dots$$

- Double DQN: since DQN tends to overestimate Q (because of the Max), use the other network to choose the optimal action

$$Q(s_t, a) = r_t + \gamma \max_a Q(s_{t+1}, a) \quad \text{Bellman} \\ Q(s_t, a) = r_t + \gamma \max_a Q'(s_{t+1}, a) \quad \text{DQN (target network)} \\ Q(s_t, a) = r_t + \gamma Q'(s_{t+1}, \text{Argmax}_a Q'(s_{t+1}, a)) \quad \text{Same formula} \\ Q(s_t, a) = r_t + \gamma Q'(s_{t+1}, \text{Argmax}_a Q(s_{t+1}, a)) \quad \text{Double DQN}$$

- Noisy networks: instead of taking actions completely at random to explore, add noise to the network weights, with learned variance (Bayesian FC layer; use a low-rank parametrization for the matrix of variances);
- Prioritized replay buffer: prioritize samples with a larger loss, *i.e.*, train more on data that surprises you (the observations are no longer iid: use sample weights);
- Dueling DQNs separately estimate the value of the state and the advantage of the action,

$$Q(s, a) = V(s) + A(s, a);$$

to ensure that the average advantage is zero, use

$$Q(s, a) = V(s) + A(s, a) - \text{Mean}_a A(s, a)$$

- Categorical (distributional) DQN: replace expectations with distributions (histograms, 51 bins).

9. To speed up the computations:

- Only compute the gradient when it is needed (not for the target or to select actions) (`detach()`);
- Only build the computation graph when it is needed (`torch.no_grad()`);
- Use larger batches: sample from several (identical) environments;
- Play and train in separate processes.

10. DQN could be used to design an intraday trading strategy: the state could be the current position and the previous 1-minute bars OHLC prices, the actions “buy” or “sell”, the reward the after-transaction-cost (realized, or unrealized) P&L. Do not expect the strategy to perform well out-of-sample.

[For some reason, all the applications of RL in finance use DQN. (My report on end-to-end portfolio construction used policy-based RL.)]

11. So far, we have focused on the value function, $V(s)$ or $Q(s, a)$, but what we really want is the policy. For small, discrete action spaces, we can easily solve the optimization problem

$$\underset{a}{\text{Maximize}} Q(s, a)$$

but for large or infinite action spaces, it is more problematic.

The **policy gradient** is

$$\begin{aligned} J &= \mathbb{E}[Q(s, a)] \\ \nabla_{\theta} J &= \mathbb{E}_{\pi_{\theta}}[Q(s, a) \nabla_{\theta} \log \pi_{\theta}(a|s)] \\ &= \mathbb{E}_{\pi_{\theta_1}}[Q(s, a) \nabla_{\theta_2} \log \pi_{\theta_2}(a|s)] \\ &= \nabla_{\theta_2} \mathbb{E}_{\pi_{\theta_1}}[Q(s, a) \log \pi_{\theta_2}(a|s)] \\ \text{Loss} &= -Q(s, a) \log \pi(a|s). \end{aligned}$$

(Note that there is no log in the definition of J , but there is one in the definition of the loss: the gradients are nonetheless the same, because the parameter θ in the expectation is fixed when we differentiate the loss, but not when we compute the expectation of J .)

The cross-entropy method is a policy gradient method with $Q(s, a) = 0$ or 1 .

REINFORCE uses the returns as Q :

$$\text{Loss} = - \sum_t G_t \log \pi(a_t|s_t).$$

However:

- The variance is *very* high: use a baseline, e.g., a constant, such as the average discounted reward, or moving average (use a **deque**), or an estimate of $V(s)$;
- Full episodes are required: we could use $Q(s, a) = r + \gamma V(s')$, with another network to estimate V (Actor-Critic), or unroll a few steps of the Bellman equation, especially if γ is not too high;
- There is not enough exploration (there is some at the beginning, because the policy is stochastic): add an *entropy bonus*, $H(\pi)$;
- The samples are correlated (and we cannot increase the replay buffer, because policy gradient methods are on-policy): use parallel environments to generate enough samples.

In practice:

- Use a baseline;
- Add an entropy bonus;
- Unroll 10 steps of the Bellman equation;
- Use parallel environments;
- Clip the gradients;
- Monitor the KL divergence between the current and previous policy (spikes are a bad sign), the L^{∞} and L^2 norm of the gradients, the components of the loss.

12. If all the Q -values are positive, the optimization of

$$\text{Loss} = -Q(s, a) \log \pi(a|s)$$

will try to move the network towards each action encountered, whether good or bad, but it will move slightly less towards bad actions. In contrast, if the Q -values are sometimes positive and sometimes negative, the optimization will move towards the good actions, and away from the bad – it will move less, and in more consistent directions.

This can be ensured with a *baseline*.

The **Advantage-Actor-Critic** (A2C) method makes the baseline state-dependent, $V(s)$:

- The *actor* computes the policy, $\pi(a|s)$;
- The *critic* computes the state value function, $V(s)$.

There are 3 losses (pay attention to their signs!): policy, value and entropy.

13. SGD works better with iid observations, but interacting with the environment gives correlated observations:

- For off-policy methods (such as DQN), use a huge replay buffer;
- For on-policy methods (such as policy gradient), use parallel environments (this is not sample-efficient).

There are two types of parallelism for RL (Asynchronous advantage actor critic, **A3C**):

- In data-level parallelism, the workers are in charge of the interactions between agent and environment, and everything else (loss, gradient, SGD) is centralized;
- In gradient-level parallelism, the workers also compute their contributions to the loss and the gradient.

14. RL (self-critical sentence training, SCST) can be used to generate text, e.g., in a chatbot, asking a customer more details about her query, and directing her to the correct page. It has several advantages over maximum likelihood estimation:

- It allows for several unrelated outputs, e.g., different translations of the same sentence, different answers to the same question – the “average” of those outputs is unlikely to be a good answer;
- We can maximize the score we want, e.g., BLEU – it does not need to be differentiable.

$$\text{MLE} : \mathbb{E}_{x \sim \text{data}} [\log p_{\text{model}}]$$

$$\text{REINFORCE} : \mathbb{E}_{\substack{s \sim \text{data} \\ a \sim \pi}} [Q(s, a) \log \pi(a|s)]$$

The rewards are too sparse to use REINFORCE directly: first pre-train with MLE (traditional seq2seq), then fine-tune with REINFORCE.

As baseline, use a greedy decoder to get a BLEU score estimate of the whole sentence.

15. The state space can be complex: for instance,

TextWorld is a grid world of which we are only given a textual description, which can be processed by an LSTM before being fed to a DQN.

16. One could also attempt to use RL to generate GUI actions from a textual description such as “Click on ‘Close’”, “Open collapsed groups and click on the link with ‘foobar’”, “select yesterday’s date using the date picker tool”, “tick checkboxes containing ‘foobar’”, etc.

17. A2C has three ingredients:

- The REINFORCE gradient,

$$\nabla J = \nabla_{\theta} \log \pi_{\theta}(a|s)(R - V_{\theta}(s));$$

- The baseline, $V_{\theta}(s)$, which minimizes the MSE with the value from the Bellman equation;
- An entropy bonus.

If the action space is continuous, represent the policy as a Gaussian distribution, parametrized by μ and σ^2 .

(Deep) Deterministic policy gradient (**DDPG**) is similar, but with a deterministic policy – so far, the policies in policy gradient (PG) methods were stochastic.

$$\nabla_{\theta} Q(s, \mu(s)) = \nabla_a Q(s, \mu(s)) \cdot \nabla_{\theta} \mu(s)$$

Note that the critic, $Q(s, a)$, is not used as in A2C: in A2C, it was optional (it was just there to reduce the variance – without it we get the Reinforce algorithm). Here, it is needed: we will train end-to-end using its gradient. Contrary to A2C, DDPG is off-policy.

Since the policy is deterministic, something else is needed to explore, e.g., adding iid noise to the actions, or using an Ornstein-Uhlenbeck process

$$dx = \theta(\mu - x)dt + \sigma dW$$

(but this requires the agent to be stateful and does not seem to improve on iid noise).

(Distributed) distributional (deep deterministic) policy gradients (**D4PG**) adds the usual DQN improvements to DDPG:

- Distribution of Q -values instead of Q -value, with the cross-entropy loss to compare the two distributions;
- n -step Bellman equation;
- Prioritized replay buffer;
- iid noise.

18. For actual robots (you can put a PyBoard, a few sensors, a few servos, on a 3D-printed frame), train them (with DDPG) in a simulator and hope the trained model also works on the robot. [Meta-learning is a more recent approach, but it requires more computation power on the robot.]

19. Small changes in the model can lead to large changes in the policy: that is why we were monitoring the KL divergence between the distribution of actions between consecutive steps. Instead of just monitoring it, trust region methods constrain it.

Proximal policy optimization (**PPO**)

- Replaces the objective $J_{\theta} = E[\log \pi_{\theta}(a_t|s_t)A_t]$ with $J_{\theta} = E[r_t \cdot A_t]$, where the importance weights are

$$r_t = \frac{\pi_{\theta_{\text{new}}}(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

$$\pi_{\theta_{\text{new}}}(a_t|s_t)/\pi_{\theta_{\text{old}}}(a_t|s_t)$$

- Clips the importance weights to $(1 - \epsilon, 1 + \epsilon)$;
- Replaces the advantage

$$A_t = -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{T-t+1} r_{T-1} + \gamma^{T-1} V(s_T)$$

with

$$A_t = \sigma_t + (\gamma\lambda)\sigma_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\sigma_{T-1}$$

where $\sigma_t = r_t + \gamma V(s_{t+1}) - V(s_t)$.

Trust region policy optimization (**TRPO**) uses the discounted visitation probability

$$\rho_{\pi}(s) = P[s_1 = s] + \gamma P[s_2 = s] + \dots,$$

defines the loss as

$$E[G] + \sum_s \rho_{\pi}(s) \sum_a \tilde{\pi}(a|s) A(s, a)$$

where G is the discounted reward, $\tilde{\pi}(a|s)$ the argmax policy, adds a constraint $KL(\theta_{\text{old}}||\theta) \leq \delta$, and uses the conjugate gradient (CG) method to solve the resulting constrained optimization problem.

ACKTR (A2C using K-FAC trust region) combines A2C, Kronecker-factored (K-FAC) second-order optimization, and trust regions.

Soft actor-critic (SAC) – currently the preferred approach for continuous control) uses

- Entropy regularization: add an entropy bonus to each reward, to encourage actions that will lead to more exploration in the long term;
- Clipped double- Q trick: learn two networks to predict the Bellman Q -value, and use the minimum, to deal with Q -value over-estimation.

There are 4 networks: π , V , Q_1 , Q_2 .

20. Population-based optimization algorithms are not sample-efficient, but eminently parallelizable.

Genetic algorithms (GA):

- Randomly perturb the current policies;
- Keep the best ones;
- Iterate.

Evolutionary strategies:

- Fit a Gaussian distribution to the parameters of the current population of policies;
- Sample from it;
- Keep the best ones;
- Iterate.

Evolutionary strategies (**CMA-ES**):

- Randomly perturb the current policy, in several ways;
- Compute the fitness of the resulting policies;

- Modify the original policy weights by adding the same noise, but rescaled proportionally to the fitness – you can use the PyTorch optimizers (not just plain descent): they just need something that looks like a gradient.

21. So far, we achieved exploration with ε -greedy policies (for value-based methods) or an entropy bonus (for policy-based methods). But taking isolated random actions may not be sufficient to explore other regions of the search space: several steps may be needed.

Noisy networks (Bayesian neural nets) add noise to the weights, with a learned variance. This may look similar to ε -greedy, but the noise is added to the weights, not the network output.

Count-based methods add a reward for rarely-visited states, c/\sqrt{N} (as UCB, for bandits). If there are many states, or if the state space is continuous, use a density estimator, or a learned state embedding, or a hashing function.

Curiosity-driven exploration does not use the reward from the environment at all, but a measure of novelty.

Prediction-based methods measure the novelty of a state by attempting to predict something from the environment: if prediction is easy, the state is not that new and does not warrant more exploration. As intrinsic reward, one could even use the ability of a (trained) neural network to predict the output from another one (untrained, randomly initialized) (more generally, the

same idea, distillation of a random neural net, can be used for outlier detection).

22. Model-based methods are more sample-efficient.

For instance, imagination-augmented agents (I2A) train an environment model $(s, a) \mapsto (r, s')$ and generate trajectories from it. They use two policies, the learned one, and a distilled version to generate trajectories, and use an RNN to compute a latent representation of the generated trajectories to add them to the input of the policy.

23. For two-player games (tic-tac-toe, chess, go), the minimax algorithm can rarely expand the full tree. Instead:

- Expand it using the current best policy (**MCTS**: Monte Carlo Tree Search); only the first moves are stochastic;
- Progressively update the value of the nodes, as in value iteration;
- Train a model on those partially expanded trees, as if they were fully expanded;
- From time to time, compare the performance of the two models (the trained one, and that used to expand the tree), and update the latter.

AlphaGo Zero uses a variant of MCTS.

24. Attempts to solve the Rubik's Cube with self-play (ADI, autodidactic iterations) were not convincing.

25. RL also applies to (competing or collaborating) multi-agent systems (MARL).

***Multiplicative interactions
and where to find them***

S.M. Jayakumar et al. (ICLR 2020)

Try replacing concatenation with *multiplication* – gating, hypernetworks, dynamic convolutions, attention, etc. can be expressed as multiplications. More generally, replace linear maps $(x, z) \mapsto W[x; z] + b$ with bilinear ones $(x, z) \mapsto z'Wx + z'U + Vx + b$ (W is then a 3-dimensional tensor).

***Gradients as features
for deep representation learning***

F. Mu et al. (ICLR 2020)

The gradient of the output wrt the parameters provides a latent representation of the input.

***Enhancing adversarial defense
by k -winners-take-all***

C. Xiao et al. (ICLR 2020)

The gradient of the loss wrt the input is used to mount adversarial attacks. One way to thwart those attacks is to use “obfuscated gradients”, *i.e.*, non-differentiable layers (differentiable wrt the parameters, for training, but not wrt the input).

But non-differentiable layers can be approximated, in the backward pass, with differentiable ones. Prefer non-differentiable layers that cannot easily be approximated by smooth functions, such as the k -winners-take-all: $\phi_k(y)_j = y_j$ if y_j is in the k largest components of y , $\phi_k(y)_j = 0$ otherwise. The loss surface, in weight space, is smooth almost everywhere (so the network is still easy to train) but in input space, it is not (so searching for adversarial examples is more problematic).

***Jacobian adversarially regularized
networks for robustness***

A. Chan et al. (ICLR 2020)

For images, the gradient of the loss wrt the input is itself an image, which can help tell how robust a model is: for robust models, it looks like the input, *i.e.*, like an actual picture, while for non-robust models, it looks like noise. During training, adding a discriminator to make this gradient image look more like a picture improves robustness (since the colours are off, add a 1-layer CNN to make the Jacobian look more like a real image).

Meta-learning with warped gradient descent

S. Flennerhag et al. (ICLR 2020)

WarpGrad learns a gradient preconditioner (as in the natural gradient, but learned, and non-linear) in the meta-learning phase – it is fixed when the final tasks are learned.

***Truth or backpropaganda? An empirical
investigation of deep learning theory***

M. Goldblum et al. (ICLR 2020)

Many commonly-held beliefs about neural networks, only proved for unrealistic architectures, such as deep linear networks, are actually wrong in general: there are sub-optimal local minima, low L^2 norm local minima need not be better, low-rank models need not be better or more robust (maximizing the rank actually works better). In spite of this, neural networks work well, thanks to their initialization.

***Piecewise linear activations substantially
shape the loss surfaces of neural networks***

F. He et al. (ICLR 2020)

ReLU networks divide the input space into “cells”; in each cell, the local minima are connected, but there are spurious minima.

***Bounds on over-parameterization
for guaranteed existence of descent paths
in shallow ReLU networks***

A. Sharifnassab et al. (ICLR 2020)

We already knew that sufficiently over-parametrized shallow ReLU networks were easy to train (the loss landscape has the “descent path property”). The converse is true: when a network is not sufficiently over-parametrized, the optimization can easily get stuck (there are “local cup minima”).

***Evaluating the search phase
of neural architecture search***

K. Yu et al. (ICLR 2020)

Evaluating network architecture search (NAS) algorithms is tricky, and they do not seem to perform much better than random search. They are biased towards fast-converging cells (shallow and wide), which do not guarantee better generalization.

NAS evaluation is frustratingly hard

A. Yang et al. (ICLR 2020)

***Understanding architectures learnt
by cell-based neural architecture search***

Y. Shu et al. (ICLR 2020)

Network architecture search (NAS) algorithms are biased towards fast-converging cells (shallow and wide), which do not guarantee better generalization.

***Understanding and robustifying
differentiable architecture search***

A. Zela et al. (ICLR 2020)

Architectures that do not generalize correspond to a sharper minimum of the loss function: stop when the largest eigenvalue of the Hessian (on a random validation minibatch) starts to increase. Alternatively, increase the regularization of the inner problem, to lower the curvature.

***Neural oblivious decision ensembles
for deep learning on tabular data***
S. Popov et al. (ICLR 2020)

Many traditional machine learning algorithms can be used as layers of deep neural networks, either as is, if they are differentiable, or after regularization if they involve discrete choices. For instance, in *oblivious decision trees* (decision trees which use the same features and thresholds at each tree level), one can use soft thresholds, *i.e.*, replace $x_i > b$ with $\sigma((\sum \lambda_i x_i - b)/s)$, and sparsemax instead of softmax for sparsity. Those layers can be stacked (NODE).

***Meta-learning acquisition functions for
transfer learning in Bayesian optimization***
M. Volpp et al. (ICLR 2020)

In GP-based Bayesian optimization, replace the acquisition function with a neural network, trained on similar objective functions; train with RL (no gradient needed).

***Learning space partitions
for nearest neighbor search***
Y. Dong et al. (ICLR 2020)

Many classical algorithms rely on some heuristic to speed them up: in particular this is the case of search algorithms (A*, DPLL, etc.), which explore a tree or a graph, and need to choose which node to explore next. Those heuristics can be replaced with a neural net.

Approximate nearest neighbour search often relies on space partition, often dependent on data (LSH, k -means (quantization), hyperplane partitions). Use supervised learning to find better partitions (neural LSH): compute the k -NN graph of the data; find a balanced partition (combinatorial graph partitioning, KaHIP); use a deep neural net to assign each point of \mathbf{R}^n to a partition. At query time, use the neural net to find the most likely bins, and search them for the nearest neighbour.

Learning to link
M.F. Balcan et al. (ICLR 2020)

Use a neural net to predict the best hyperparameters (of hierarchical clustering).

***Neural network branching
for neural network verification***
J. Lu and M.P. Kumar (ICLR 2020)

Use a graph neural net (GNN) to find branch-and-bound heuristics.

Explanation by progressive exaggeration
S. Singla et al. (ICLR 2020)

Counterfactuals are a way of explaining the forecasts of a model: minimal changes to the input to change the output (adversarial examples). They are usually

computed as an optimization problem, but one could use a neural network to automatically compute them.

The counterfactual explanation should look real: use a discriminator. Also add a self-consistency loss (if the output is the age of a person from a picture, making it 10 years older, and then 10 years younger, should recover the initial image).

Mathematical reasoning in latent space
D. Lee et al. (ICLR 2020)

There are three main types of logic: propositional logic (and, or, not operators, but no quantifiers: SAT solvers, such as minisat), first-order logic (with quantifiers over variables: SMT solvers, e.g., z3), and higher-order logic (with quantifiers over variables, functions and propositions: theorem provers, such as coq, agda, isabelle). HOL Light is an interactive theorem prover, with a database of 19,000 mathematical statements, covering algebra, topology, calculus, and measure theory. Like most theorem provers, it relies on rewrite rules – but it is very time-consuming to check which rules can be applied, and where they lead to.

With a graph neural net, one can learn an embedding that helps tell if a rewrite rule applies to a given statement (by taking the scalar product of the embeddings) and perform approximate reasoning, by moving several steps in the latent space.

***Learning heuristics for quantified boolean
formulas through reinforcement learning***
G. Lederman et al. (ICLR 2020)

Learned constraint solver heuristics outperform hand-crafted ones. A formula in conjunctive normal form (CNF) can be represented as a bipartite graph, whose nodes are terms and variables, which can be fed to a graph neural net (GNN).

***Deep symbolic superoptimization
without human knowledge***
H. Shi et al. (ICLR 2020)

To simplify expressions, in a compiler, one can use reinforcement learning to select which rules to apply among a set of predefined rewrite rules. To avoid the dependence on rules provided by humans, model the expressions as trees, compute node (or rather sub-tree) embeddings (with a tree LSTM), select a subtree (with a neural network), decode it using a tree LSTM with attention, replace the subtree with the new one and start again. Train with Reinforce and curriculum learning with increasingly long expressions.

***CLN2INV: learning loop invariants
with continuous logic networks***
G. Ryan et al. (ICLR 2020)

Automated program verification struggles with loops: the loop invariants still have to be specified by humans – once provided, the SMT solver can do its job.

One can use neural networks to guide the search for invariant formulas from recorded execution traces, after relaxing logical formulas into continuous logic networks (CLN) (and progressively tightening the relaxation during the optimization).

***Oblique decision trees
from derivatives of ReLU networks***
G.H. Lee and T.S. Jaakkola (ICLR 2020)

An *oblique decision tree* is a decision tree whose decision nodes are linear classifiers (*i.e.*, of the form $w'x < \alpha$ instead of $x_i < \alpha$). Locally constant neural networks model oblique decision trees, in a parsimonious way (N neurons suffice for 2^N nodes); they can be obtained as the gradient of a ReLU network, further processed with another neural net.

***PCMC-Net:
feature-based pairwise choice markov chains***
Alix Lhéritier (ICLR 2020)

Traditional *choice models* (multinomial logit, random utility models) do not account for human idiosyncrasies: independence of irrelevant alternatives, regularity (increasing the choice set does not change the ranking). *Pairwise choice Markov chain* (PCMC) models are more general: they use a Markov chain whose transition matrix models the preference between pairs of alternatives and consider its stationary distribution. If there are too many alternatives, use a neural net to approximate the transition matrix and compute the stationary distribution. Application: airline itinerary choice

***Higher-order function networks for learning
composable 3D object representations***
E. Mitchell et al. (ICLR 2020)

Hypernetworks are neural networks computing the weights of another network. They can be used to design input-dependent networks, for instance to apply a different decoder for each type of object when reconstructing it from a cloud of points.

***Neural epitome search
for architecture-agnostic network compression***
D. Zhou et al. (ICLR 2020)

Hypernetwork to uncompress, on the fly, the weights of a neural network.

***You only train once:
loss-conditional training of deep networks***
A. Dosovitskiy and J. Djolonga (ICLR 2020)

Hypernetworks can be used to generate a family of image compression models, using

$$\text{compression} + \lambda \cdot \text{reconstruction}$$

as loss, with the weights depending on λ .

Continual learning with hypernetworks
J. von Oswald et al. (ICLR 2020)

Hypernetworks can also be used for continual learning, preventing catastrophic forgetting by computing an embedding of the task, computing the new weights from that embedding, with a constraint that the weights of the previous tasks to remain fixed.

***Generalization bounds
for deep convolutional neural networks***
P.M. Long and H. Sedghi (ICLR 2020)

Neural networks may appear too expressive to lead to good generalization, but the training procedures do not stray far from the initialization, considerably reducing the risk of overfitting. Increasing the number of parameters decreases the distance from initialization.

***Learning to balance: Bayesian meta-learning
for imbalanced and out-of-distribution tasks***
H.B. Lee et al. (ICLR 2020)

The goal of meta-learning, or “learning to learn”, is to be able to learn a new task with little data (*few-shot learning*) and only a few steps of gradient descent. MAML (model-agnostic meta learning) is one of the popular algorithms for this: it learns a good initialization, from which new tasks can be learned quickly.

To account to task imbalance, class imbalance and distribution shift, in meta-learning (MAML),

- Use sample and task statistics to recognize these;
- Feed them to a neural net to compute the step size and number of steps (more and smaller steps for larger tasks – the gradients are larger and there is more information to extract), and the shift to apply to the initial parameters (to account for distribution shift).

***Meta dropout: learning to perturb
latent features for generalization***
H.B. Lee et al. (ICLR 2020)

To improve the decision boundaries in few-shot learning (MAML), learn input-dependent data augmentation (multiplicative noise, at each layer).

***Empirical Bayes transductive meta-learning
with synthetic gradients***
S.X. Hu et al. (ICLR 2020)

MAML learns a weight initialization that can be used for any new task. Instead, leverage the unlabeled data of the new task to choose that initialization.

***Rapid learning or feature reuse? towards
understanding the effectiveness of MAML***
A. Raghu et al. (ICLR 2020)

MAML uses an outer loop to find a good initialization, and an inner loop, on a few samples, to adapt that

initialization to the new task. Comparing the representations learned across several tasks shows that only the last layer changes: MAML relies on feature reuse. To speed things up, only train the last layer in the inner loop (ANIL).

Transferring optimality across data distributions via homotopy methods

M. Gargiani et al. (ICLR 2020)

In transfer learning, progressively deform the source task to the target task (with a simple convex combination).

Cross-domain few-shot classification via learned feature-wise transformation

H.Y. Tseng et al. (ICLR 2020)

In few-shot classification, try to make the feature distribution, for the new task, more diverse, with a “feature-wise transformation layer”.

Scalable and order-robust continual learning with additive parameter decomposition

J. Yoon et al. (ICLR 2020)

In continual learning, separate shared parameters from task-specific parameters.

Sliced Cramer synaptic consolidation for preserving deeply learned representations

S. Kolouri et al. (ICLR 2020)

To prevent catastrophic forgetting in sequential learning, one could keep track of some of the inputs of the previous tasks, and add a penalty to ensure their latent representations are kept. Instead, try to preserve the *distribution* of those latent representations.

An exponential learning rate schedule for deep learning

Z. Li and S. Arora (ICLR 2020)

There are countless learning rate schedules: decreasing, triangular, cosine, etc. Even weirder ones work, such as an exponentially increasing one (yes, increasing, not decreasing: multiply the learning rate by $1+c$, $c > 0$, at each iteration) – this assumes the presence of normalizing layers, which make the loss function scale-invariant.

The tapered exponential schedule has an exponential growth in each phase, and a small drop at the end of each phase, before moving to a lower growth rate.

Budgeted training: rethinking deep neural network training under resource constraints

M. Li et al. (ICLR 2020)

If your training budget is limited and known in advance, linearly decreasing the learning rate, down to zero, may be good enough.

Rethinking the hyperparameters for fine-tuning

H. Li et al. (ICLR 2020)

What matters is not really the learning rate, but the *effective learning rate*, $\eta/(1-m)$, which combines learning rate and momentum.

Revisiting self-training for neural sequence generation

J. He et al. (ICLR 2020)

Semi-supervised learning often works as follows: train a teacher network on real data; train a student network on data labeled by the teacher; fine-tune the student on real data; iterate (the student becomes teacher).

Dropout is important (noisy self-training).

Contrastive representation distillation

Y. Tian et al. (ICLR 2020)

Distillation is similar, but instead of matching the output of the teacher network, *i.e.*, the class labels, the student network tries to match the logits. Instead, one could try to match the features, one layer before, after mapping them to the same vector space (they have different dimensions), using the cosine distance.

DeepHoyer: learning sparser neural network with differentiable scale-invariant sparsity measures

H. Yang et al. (ICLR 2020)

Instead of the lasso sparsifying regularizer, try the **Hoyer-square** regularizer $H(w) = \|w\|_1^2 / \|w\|_2^2$ (or the group Hoyer). It is differentiable (like L^1) and scale-invariant (like L^0); it still trims small values, but preserves large ones.

Padé activation units: end-to-end learning of flexible activation functions in deep networks

A. Molina et al. (ICLR 2020)

The activation functions (ReLU, sigmoid, tanh, etc.) are usually fixed. Instead, we can choose a flexible parametrization, that allows a large variety of shapes, and learn the activation functions. The *Padé approximations*, *i.e.*, rational functions (quotients of polynomials)

$$g_{\theta}(x) = \frac{\sum_{i=0}^m \theta_i x^i}{1 + \left| \sum_{j=1}^n \theta_{m+j} x^j \right|}$$

are often used by computers in numerical computations (e.g., for special functions). A random variant adds a small uniform noise to θ .

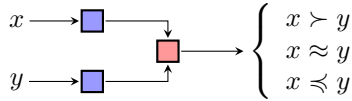
Effect of activation functions on the training of overparametrized neural nets

A. Panigrahi et al. (ICLR 2020)

Most theoretical studies assume ReLU activations. Experiments suggest that non-smooth activation functions (e.g., ReLU, ELU) lead to faster convergence (than swish, tanh). The situation improves with depth.

**Order learning
and its application to age estimation**
K. Lim et al. (ICLR 2020)

Here is another way of adding (robustness to) nonlinearities to a neural net: do not output an absolute value (age, distance, pitch, score, etc.), but compare inputs, with a siamese network; at test time, the numeric value can be computed by interpolation.



This approach can be combined with clustering, if there are distinct groups in the input requiring different models: train separate models for random clusters, update cluster membership from the model performance, iterate.

**Geometric analysis
of nonconvex optimization landscapes
for overcomplete learning**
Q. Qu et al. (ICLR 2020)

Overcomplete dictionary learning looks for $A \in \mathbf{R}^{n \times m}$ and $X \in \mathbf{R}^{m \times p}$ such that $Y = AX$. Since the problem has many symmetries, it is non-convex, but if all minima are equivalent, and if the saddle points are not flat, it can be solved efficiently. We can require A to be row-orthonormal, and column-near-orthogonal, *i.e.*, impose an upper bound on the absolute value of the correlation (*i.e.*, cosine) of any two columns. To get one column of A , solve

$$\begin{array}{ll} \text{Find} & q \in \mathbf{S}^{n-1} \\ \text{To minimize} & -\|q'Y\|_4^4 \end{array}$$

(in the over-complete case, use the spikiness, L^4 , rather than L^1).

With pre-conditioning, this can be generalized to *convolutional dictionary learning* (A is no longer near-orthogonal): $y_i = \sum_k a_k \otimes x_{ki}$, where $y_i, a_k, x_{ki} \in \mathbf{R}^n$, with x_{ki} sparse

Examples include phase retrieval, low-rank matrix recovery, phase synchronization, shallow/linear neural net, sparse blind deconvolution, etc.

**Short and sparse deconvolution:
a geometric approach**
Y. Lau et al. (ICLR 2020)

**Understanding l^4 -based dictionary learning:
interpretation, stability, and robustness**
Y. Zhai et al. (ICLR 2020)

The L^4 norm, or “spikiness”, can be used instead of the L^2 or L^1 norms, for instance, for dictionary learning:

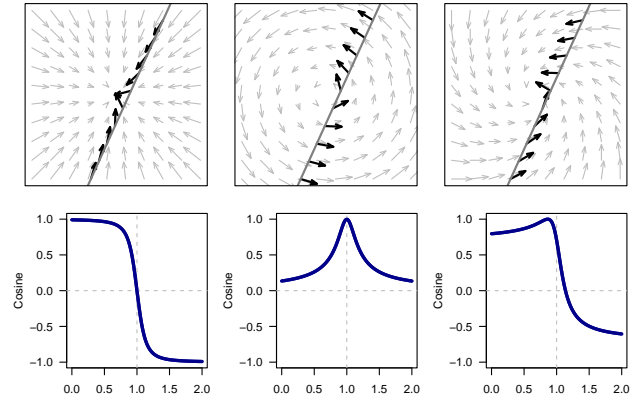
$$\begin{array}{ll} \text{Find} & A \in O_n \\ \text{To maximize} & \|AY\|_4^4 \end{array}$$

**A closer look at the optimization landscapes
of generative adversarial networks**
H. Berard et al. (ICLR 2020)

Consider the cosine similarity between the *game vector field* v , *i.e.*, the concatenation of the gradients of the generator and discriminator, and the linear path $(\theta_t)_{t \in [0,1]}$ between the initial and final parameters.

$$c_t = \cos(\theta_1 - \theta_0, v(\theta_t))$$

Plotting c_t (and $\|v(\theta_t)\|$) for $t \in [0.5, 1.5]$ reveals two possible behaviours: attraction (decreasing) and rotation (bump at 1).



**An inductive bias for distances: neural nets
that respect the triangle inequality**
S. Pitis et al. (ICLR 2020)

When learning a distance function, we want it to satisfy the triangle inequality. Deep metric learning often learns an embedding and then computes the Euclidean distance in that latent space – but not all metrics can be computed in that way.

A *DeepNorm* network is

$$\begin{array}{l} h_i = g_i(W_i h_{i-1} + U_i x) \\ \|x\| = h_k \end{array}$$

where $h_0 = 0$, $W_1 = 0$, the W_i are non-negative, g is convex and positive homogeneous, *e.g.*,

$$\begin{array}{l} g(u) = \alpha \text{relu}(u) + \beta u \\ \max \text{ReLU}(x, y) = [\max(x, y), \alpha \text{relu}(x) + \beta \text{relu}(y)] \end{array}$$

where $\alpha, \beta \geq 0$.

Wide Mahalanobis norms are of the form

$$\|x\| = \text{MaxMean}_i(\|W_i x\|_2)$$

$$\text{MaxMean}(x_1, \dots, x_n) = \alpha \text{Max}(x_1, \dots, x_n) + (1 - \alpha) \text{mean}(x_1, \dots, x_n).$$

Deep batch active learning by diverse, uncertain gradient lower bounds
J.T. Ash et al. (ICLR 2020)

Active learning approaches can be uncertainty-based (for small batches) or diversity-based (for larger batches). They can be combined: use the gradient of the loss (for the most likely label) wrt the parameters as a representation of the unlabeled examples, and sample using a determinantal point process (DPP) or (less expensive) k -means++.

Accelerating SGD with momentum for over-parameterized learning
C. Liu and M. Belkin (ICLR 2020)

Nesterov momentum provably accelerates (full) gradient descent, but not stochastic gradient descent. Instead, try

$$\begin{aligned} w_{t+1} &\leftarrow u_t - \eta_1 \nabla f(u_t) \\ u_{t+1} &\leftarrow (1 + \gamma) w_{t+1} - \gamma w_t + \eta_2 \nabla f(u_t) \end{aligned}$$

Variance reduction with sparse gradients
M. Elibol et al. (ICLR 2020)

The SpiderBoost gradient,

$$g_{n+1} = g_n + \nabla f_n(\theta_n) - \nabla f_n(\theta_{n-1}),$$

where f_n is the loss on the n th batch, can be approximated with sparse updates, only computing a random subset of the coordinates with the largest variance (sparse forward and backward passes).

Coherent gradients: an approach to understanding generalization in gradient descent-based optimization
S. Chatterjee (ICLR 2020)

Models trained with gradient descent (GD) generalize well (even though they could just memorize their input) because strong gradient directions (common to many observations) are kept and weak gradients cancel out. It is possible to perturb GD to suppress weak gradients (cf. random forests).

Stochastic weight averaging in parallel: large-batch training that generalizes well
V. Gupta et al. (ICLR 2020)

Start training with large batches, but stop early; then, train on small batches, in parallel; finally average the weights.

Why not to use zero imputation? correcting sparsity bias in training neural networks
J. Yi et al. (ICLR 2020)

Replacing missing values with 0 changes the scale of the output of the first layer; rescale the input accordingly (as with dropout).

ProxSGD: training structured neural networks under regularization and constraints
Y. Yang et al. (ICLR 2020)

Although the L^1 norm is a sparsifying constraint, SGD cannot push the weights to exactly zero – it cannot enforce hard constraints. Replace the gradient step

$$\begin{aligned} g &\leftarrow \nabla_x f(x) + r(x) \\ \hat{x} &\leftarrow \hat{x} - \alpha g \end{aligned}$$

with a proximal step

$$\begin{aligned} g &\leftarrow \nabla_x f(x) \\ \hat{x} &\leftarrow \underset{x \in X}{\text{Argmin}}(x - \hat{x})' g + \frac{1}{2} \|x - \hat{x}\|^2 + r(x) \end{aligned}$$

When the penalty r is the L^1 norm, the proximal operator is soft-thresholding.

Identity crisis: memorization and generalization under extreme overparameterization
C. Zhang et al. (ICLR 2020)

To study how overparametrized models generalize, look at extreme situations, such as learning with just one sample (learning the constant map) and learning the identity map. Shallow networks are better at learning the identity map, deeper networks are better at learning the constant map.

Deep double descent: where bigger models and more data hurt
P. Nakkiran et al. (ICLR 2020)

The *double descent* phenomenon is visible for:

- test error vs model size;
- test error vs epoch.

Gradient descent maximizes the margin of homogeneous neural networks
K. Lyu and J. Li (ICLR 2020)

Deep neural nets are too expressive and should overfit. They work because gradient methods have an implicit bias towards maximum-margin solutions.

Unbiased contrastive divergence algorithm for training energy-based latent variable models
Y. Qiu et al. (ICLR 2020)

Contrastive divergence (CD), used to train energy-based models (EBM) such as restricted Boltzman machines (RBM), uses a biased gradient estimator (a small, fixed, number of MCMC steps) and may diverge. *Unbiased MCMC* can provide an unbiased gradient estimator (by using a random number of MCMC steps).

Mixout: effective regularization to finetune large-scale pretrained language models
C. Lee et al. (ICLR 2020)

Drop-connect sets some of the weights to zero. Instead, set them to the weights of a pre-trained network.

Self-supervised learning of appliance usage
C.Y. Hsu et al. (ICLR 2020)

To detect causality in weakly-related data streams, such as total energy consumption of a home over time, and people's location in the home (approximate, from Wifi perturbations), try to predict one from the other (the cause from the consequence: location given energy event): when prediction is possible, the two are related.

SELF: learning to filter noisy labels with self-ensembling
D.T. Nguyen et al. (ICLR 2020)

With low-quality (crowd-sourced) training data, some of the labels may be incorrect. Only train on reliable labels: to identify them, compare the predictions of the current model and that a few epochs ago: more reliable labels have more stable predictions.

DivideMix: learning with noisy labels as semi-supervised learning
J. Li et al. (ICLR 2020)

To identify noisy observations, one can model that noise: the per-sample loss is then a two-component mixture (noisy samples have a larger loss). One can then discard the noisy labels and use semi-supervised learning. To avoid confirmation bias, train two networks, each filtering the other's noise.

Simple and effective regularization methods for training on noisily labeled data with generalization guarantee
W. Hu et al. (ICLR 2020)

To deal with label noise, one could add a penalty for the distance to the initialization parameters, or actually model the noise by adding trainable auxiliary variables. For a wide neural net, these are equivalent to kernel *ridge* regression with a neural tangent kernel.

AugMix: a simple data processing method to improve robustness and uncertainty
D. Hendrycks et al. (ICLR 2020)

Traditional data augmentation applies one transformation to the image. Instead, apply several, and mix

the results (with a weighted DAG); add a consistency penalty to the loss to ensure that different augmentations yield the same result.

ReMixMatch: semi-supervised learning with distribution matching and augmentation anchoring
D. Berthelot et al. (ICLR 2020)

MixMatch performs semi-supervised learning by training on the labeled data, running the model on many augmentations of the unlabeled data, and using the resulting label distribution. ReMixMatch enforces consistency among augmentations, makes the label distribution close to that of the training set, and adds a self-supervised loss (rotation prediction).

Robust training with ensemble consensus
J. Lee and S.Y. Chung (ICLR 2020)

To make a network more robust, we can train it on the $(100 - e)\%$ observations with the smallest loss (as with a truncated regression).

Identify noisy examples among those small-loss examples by perturbing the network: if an image was not noisy (learned by generalization), the prediction does not change, but if it was noisy (learned by memorization), it changes.

An ensemble of neural nets, obtained by adding noise, at test time, to the input or to the network (e.g., with dropout), or by taking the same network from previous epochs (a "self-ensemble") is more robust than a single network.

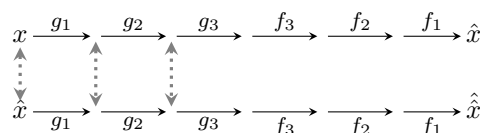
Self-labelling via simultaneous clustering and representation learning
Y.M. Asano et al. (ICLR 2020)

Jointly learn a representation and a clustering (or labeling) of the data, with a constraint that each label be used the same number of times, by iteratively computing the optimal labeling (optimal transport) and training a classifier.

Use data augmentation. You can learn several clusterings (classification heads – cf. DeepCluster) at the same time.

RaPP: novelty detection with reconstruction along projection pathway
K.H. Kim et al. (ICLR 2020)

Autoencoders can be used to detect anomalies: their reconstruction error is large. Do not only compare the input and the output of the network, feed the output back to the input, and also compare the intermediate layers: if the autoencoder is



also compare $g_1(x)$ and $g_1(\hat{x})$, $g_2(g_1(x))$ and $g_2(g_1(\hat{x}))$ etc.

***Robust subspace recovery layer
for unsupervised anomaly detection***
C.H. Lai et al. (ICLR 2020)

Auto-encoders can be used for anomaly detection: the reconstruction error is larger than usual. But auto-encoders are not robust to outliers: add a *robust subspace recovery* layer $z \mapsto Az$ between the encoder and the decoder, which projects to an even lower-dimensional subspace, with $\|AA' - I\|$ and $\|z - A'Az\|_2$ as penalties.

***Classification-based anomaly detection
for general data***
L. Bergman and Y. Hoshen (ICLR 2020)

A similar idea can be applied to many other models: for out-of-distribution samples, the error is often larger. For instance, GOAD picks M random affine transformations, transforms the data with them, and tries to recover which transformation was used: the prediction accuracy is an anomaly score.

***Input complexity
and out-of-distribution detection
with likelihood-based generative models***
J. Serrà et al. (ICLR 2020)

Likelihood can be used to identify out-of-distribution datasets – but the distribution of those likelihoods can be higher than for the training set (it is easier), or lower (more difficult). The likelihood is correlated with the complexity of the data (Kolmogorov complexity, approximated by a lossless compression algorithm). Compensate for this relation:

$$S(x) = -\text{LogLik}(x) - \text{Complexity}(x).$$

This can be interpreted as a likelihood ratio test.

Novelty detection via blurring
S. Choi and S.Y. Chung (ICLR 2020)

Random network distillation (RND) takes two networks: a randomly-initialized, fixed network g , and a network f trained to replicate g on the training data. Since they only match on the training data, their difference $\|f(x) - g(x)\|$ is larger for out-of-distribution samples. Blurring the data (RND trained on both raw and blurred data) helps.

White noise analysis of neural networks
A. Borji and S. Lin (ICLR 2020)

Titration (progressively adding noise to images and checking the output, or the confusion matrix) can help understand classification biases

***Feature interaction interpretability: a case
for explaining ad-recommendation systems
via neural interaction detection***
M. Tsang et al. (ICLR 2020)

Detect interactions via feature perturbation, and use them for sparse feature crossing.

Detecting extrapolation with local ensembles
D. Madras et al. (ICLR 2020)

To see if a model is extrapolating (and if the forecasts are unreliable), train several models (with the same structure) and check if they agree. This can be done with a single model as well: at the local minimum, changing the weights in the direction of the smallest eigenvalues of the Hessian of the loss does not change the loss much, and provides a “self-ensemble” of models.

***Conservative uncertainty estimation
by fitting prior networks***
K. Ciosek et al. (ICLR 2020)

To estimate the uncertainty of a neural network, train several networks, with different random initializations, and check if they match.

***Towards neural networks
that provably know when they don't know***
A. Meinke and M. Hein (ICLR 2020)

Our models do not really model $P[y|x]$, but $P[y|x, \text{in-distribution}]$. We can actually estimate $P[y|x]$ with Bayes's formula.

***Quantifying point-prediction uncertainty
in neural networks
via residual estimation with an i/o kernel***
X. Qiu et al. (ICLR 2020)

Given a trained neural net, train another neural net to predict the uncertainty of the forecasts

$$(x, \hat{y}) \mapsto \|y - \hat{y}\|^2.$$

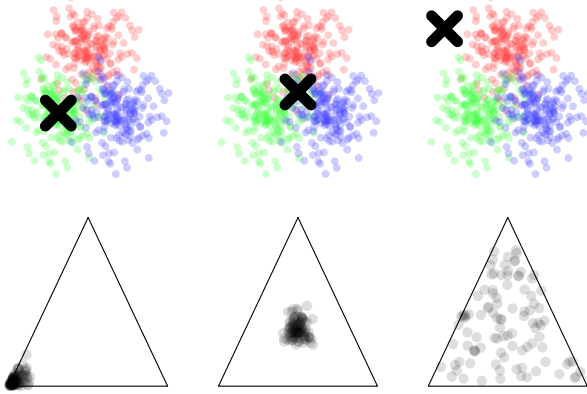
***Distance-based learning from errors
for confidence calibration***
C. Xing et al. (ICLR 2020)

To estimate the uncertainty of a prediction from a classifier, cluster the observations in each class, and use the distance to the cluster center. Use a neural network to approximate those uncertainty estimates.

Ensemble distribution distillation
A. Malinin et al. (ICLR 2020)

There are two types of uncertainty:

- Data uncertainty: the classes overlap, or we are too close to the decision boundary;
- Knowledge uncertainty: out-of-distribution observation, *i.e.*, we are too far from the training data.



The situation can be estimated from an ensemble of models:

- If the models are all unsure, we are close to the decision boundary;
- If the models are sure but disagree, the data is out-of-distribution;
- If the models are sure and agree, that is fine.

Do not distill a single teacher model into a student one, nor the mean of an ensemble of teacher models, but both mean and diversity of the ensemble, using Dirichlet distributions.

A constructive prediction of the generalization error across scales
J.S. Rosenfeld et al. (ICLR 2020)

Predict the generalization error by fitting a model

$$\text{test error} \sim \log(\#\text{samples}) + \log(\#\text{parameters}).$$

Pitfalls of in-domain uncertainty estimation and ensembling in deep learning
A. Ashukha et al. (ICLR 2020)

Before comparing (ensembles of) models (with the log-likelihood, or the Brier score), ensure the models are calibrated. To assess an ensembling technique (which may produce dependent models), compute the equivalent number of independent models (for instance, an ensemble of 100 dropout variants of a models could be equivalent to 2 independently-trained models). Test-time augmentation works well.

Unrestricted adversarial examples via semantic manipulation
A. Bhattad et al. (ICLR 2020)

Computer vision neural networks can be trained to be robust to adversarial attacks, in the sense that images close to the input will give the same output, where closeness is measured with the L^∞ or L^2 norm.

But this leaves enough room for photo-realistic attacks, manipulating colour and texture or adding barely perceptible shadows.

Breaking certified defenses: semantic adversarial examples with spoofed robustness certificates
A. Ghiasi et al. (ICLR 2020)

The robustness certification radius is not a good measure of robustness: we can find attacks that look like natural images, get misclassified, but have a large certification radius. The shadow attack changes the luminosity of each pixel, as if there were a shadow on the image; add a penalty to make it smooth (no edges), and another to keep the change small.

A target-agnostic attack on deep models: exploiting security vulnerabilities of transfer learning
S. Rezaei and X. Liu (ICLR 2020)

Many models are built from publically-available pre-trained networks: only the head of the network is unknown to the attacker. Inputs with a single very large activation provide good adversarial attacks.

Transferable perturbations of deep feature distributions
N. Inkawhich et al. (ICLR 2020)

Using the latent feature distributions, from the intermediate layers, leads to more transferable attacks.

Thieves on sesame street! model extraction of BERT-based APIs
K. Krishna et al. (ICLR 2020)

If a model, trained by fine-tuning a publically-available pre-trained model (e.g., BERT – the availability of the pre-trained model is important) on proprietary data, has a public API (e.g., Google Translate), it is (economically) possible to extract it, by feeding it data (e.g., random words, with no grammatical structure, or Wikipedia excerpts), collecting the results, and training a new model. This model stealing attack allows intellectual property theft, (white-box) adversarial attacks, and data leakage.

Defences include API watermarking (for 0.1% of the queries, return an incorrect answer and store the input/answer pair) and membership classification (recognize out-of-distribution inputs and return random data), but they only work on naive adversaries.

Prediction poisoning: towards defenses against DNN model stealing attacks
T. Orekondy et al. (ICLR 2020)

To protect a trained model $f : x \mapsto y$ against stealing attacks, try to perturb the attacker's gradient: do not return $y = f(x)$, but $y + \delta$, where δ is small, $|\delta| < \varepsilon$ and maximizes the angle between the gradients $\nabla \text{Loss}(y)$ and $\nabla \text{Loss}(y + \delta)$.

EMPIR: ensembles of mixed precision deep networks for increased robustness against adversarial attacks
S. Sen et al. (ICLR 2020)

Low-precision networks are less accurate but more robust to adversarial attacks than full-precision ones: use an ensemble of both.

Enhancing transformation-based defenses against adversarial attacks with a distribution classifier
C. Kou et al. (ICLR 2020)

To defend against adversarial attacks, one can use data augmentation at test time, *i.e.*, feed several altered images to the classifier, and use a majority vote – but accuracy degrades. Instead, train the model that will actually be used, the model including the data augmentation.

Defending against physically realizable attacks on image classification
T. Wu et al. (ICLR 2020)

Training against rectangular occlusion attacks improves robustness to physically realizable attacks.

Skip connections matter: on the transferability of adversarial examples generated with resnets
D. Wu et al. (ICLR 2020)

Replace the skip connection $z_1 = z_0 + f(x_0)$ gradient $1 + f'$ with $1 + \gamma f'$, $\gamma < 1$: the resulting network is more robust to transferred (black-box) adversarial attacks.

Adversarial Lipschitz regularization
D. Terjék (ICLR 2020)

Lipschitz regularization is usually implemented with a gradient penalty. The Lipschitz regularization, $E[|f(y) - f(x)| / |y - x|]$, tends to diverge when estimated using pairs of points from a minibatch: instead, use adversarial examples, $y = x + r$, $|r| \leq \varepsilon$ (one could also use a gradient penalty, as in WGANs).

Consistency regularization for generative adversarial networks
H. Zhang et al. (ICLR 2020)

The gradient penalty (Lipschitz) adds a computational overhead. Use *consistency regularization* (as in semi-supervised learning – deformed images should have the same output) on the discriminator instead of the Lipschitz constraint (gradient penalization, spectral normalization) (CR-GAN)

Intriguing properties of adversarial training at scale
C. Xie et A. Yuille (ICLR 2020)

Removing clean images from the training set makes the model more robust to adversarial attacks. Clean

images and adversarial examples seem to come from different distributions: in the BatchNorm layers, use different statistics for the clean and adversarial inputs (mixture batch normalization), or use a non-batch normalization (e.g., group norm).

Deeper networks are more robust.

Universal approximation with certified networks
M. Baader et al. (ICLR 2020)

Interval analytics can provide ℓ^∞ robustness certification (robustness to ℓ^∞ adversarial examples) but, for arbitrarily-trained networks, the intervals are often too large. However, for any function, there exists a ReLU network approximating it and whose ℓ^∞ robustness can be proved by interval analysis.

Adversarially robust representations with smooth encoders
T. Cemgil et al. (ICLR 2020)

VAEs are not robust to adversarial perturbations; the encoder is to blame: train it adversarially. The adversarial training searches for an input image, close to the initial input, but with latent representation as far away (for the Wasserstein distance) as possible. To check that the output is also robust, use a classifier (detecting the presence of glasses, of a smile, etc.) and try to change the predicted class.

Sign bits are all you need for black-box attacks
A. Al-Dujaili and U.M. O'Reilly (ICLR 2020)

Most attacks require an estimate of the gradient of the network wrt to its input, which requires $O(n)$ queries. Looking for the sign of the gradient (an element of $\{\pm 1\}^n$ instead of \mathbf{R}^n) still requires $O(n)$ queries, but an adversarial attack is often found much earlier.

Sign-OPT: a query-efficient hard-label adversarial attack
M. Cheng et al. (ICLR 2020)

Understanding the limitations of conditional generative models
E. Fetaya et al. (ICLR 2020)

Discriminative models are not robust to outliers and adversarial attacks. Generative models do not necessarily fare better: with MNIST digits on CIFAR background, images interpolated between digits have a higher log-likelihood (because there is more content in the background, more entropy unrelated to the class).

Data-independent neural pruning via coresets
B. Mussay et al. (ICLR 2020)

Prune the weights of a neural network by keeping those with the highest importance (if there are several outputs, take the maximum of the importances) and then fine-tuning.

***Provable filter pruning
for efficient neural networks***
L. Liebenwein et al. (ICLR 2020)

Pruning is usually empirical (no guarantees), and only looks at the weights, not the data. Instead, look at the contributions of a neuron to the activations of the next layer.

***Lookahead: a far-sighted alternative
of magnitude-based pruning***
S. Park et al. (ICLR 2020)

Magnitude-based pruning (discarding low weights) does not take into account the connectivity of the network: if a large-weight edge is only connected to small-weight edges, it may not be that important, and could also be discarded – taking into account the previous and next layer can change which edges seem important: prune weights that are small and/or connected to small weights. More formally, do not prune W_k into \tilde{W}_k to minimize $\|W_k x - \tilde{W}_k x\|_2$ but to minimize $\|W_{k+1} W_k W_{k-1} x - W_{k+1} \tilde{W}_k W_{k-1} x\|_2$.

***Comparing rewinding and fine-tuning
in neural network pruning***
A. Renda et al. (ICLR 2020)

After pruning a network, one usually fine-tunes it, but, since the network is now different, the optimal hyperparameters are no longer the same: they have to be re-estimated. *Rewinding* the weights (*i.e.*, using old weights) or just the learning rate, before fine-tuning, works as well as hyperparameter tuning.

***The intriguing role of module criticality
in the generalization of deep networks***
N.S. Chatterji et al. (ICLR 2020)

Rewind a layer (only one) back to its initialization: the impact is sometimes visible, sometimes not. This suggests that some layers are more important than others; this can be explained by the loss landscape, the path from the initialization to the final weights, which looks like a tunnel for some layers (less important), or like a funnel (more important). The width of that funnel (network criticality) is related to generalization.

Dynamic model pruning with feedback
T. Lin et al. (ICLR 2020)

Pruning can be done before training, after training, several times during training, or dynamically during training, potentially reactivating prematurely pruned weights). This can be done with a binary mask on the weights (use the mask in the forward pass, but not in the backward pass; the mask can change at each step) or a threshold vector (layer-wise, filter-wise or neuron-wise).

***Dynamic sparse training:
find efficient sparse network from scratch
with trainable masked layers***
J. Liu et al. (ICLR 2020)

Instead of training, pruning and fine-tuning a network, learn both the weights and a threshold vector (layer-wise, filter-wise or neuron-wise).

***Signal propagation perspective for pruning
neural networks at initialization***
N. Lee et al. (ICLR 2020)

Surprisingly, it is possible to prune the network before training, by looking at the change in loss when removing a parameter, which can be computed as a gradient, $\nabla_c \text{Loss}(c \odot w)$ evaluated at $c = \mathbf{1}$, where $c \in \{0, 1\}^m$ is the mask. This works better if the singular values of the weights matrices of each layer (strictly speaking, the singular values of the jacobian of each layer) are close to 1.

***One-shot pruning of recurrent neural networks
by jacobian spectrum evaluation***
S. Zhang and B.C. Stadie (ICLR 2020)

For efficient backward/forward propagation in a RNN, we want the eigenvalues of $J_t = \partial h_t / \partial h_{t+1}$ to be close to 1, but they are initially small: only keep the weights likely to help maximize $\chi = \sum_t \mathbb{E} \|J_t\|_F$, *i.e.*, those with the largest value of $|\partial_\theta \chi|$ (set the others to 0, permanently). (the weights are pruned before training, the first time they see the data: it is a sparse initialization)

***Batch-shaping for learning
conditional channel gated networks***
B.E. Bejnordi et al. (ICLR 2020)

To reduce the energy consumption of a neural net, only use part of it, with a gating mechanism to choose which features to compute.

Depth-adaptive transformer
M. Elbayad et al. (ICLR 2020)

Sequence-to-sequence models have more and more parameters; they can be shrunk by distillation or structured pruning. Instead, reduce the model in a sample-specific way: add exits at each level of the decoder, and a halting mechanism.

***Triple wins: boosting
accuracy, robustness and efficiency together
by enabling input-adaptive inference***
T.K. Hu (ICLR 2020)

Multi-output networks seem more robust to adversarial attacks.

***Gradient L^1 regularization
for quantization robustness***
M. Alizadeh et al. (ICLR 2020)

Neural networks can be trained to be more robust to post-training quantization (a bounded, additive perturbation) by adding an L^1 penalty for the gradient of the loss (wrt the weights).

Robust learning with jacobian regularization
J. Hoffman et al. (ICLR 2020)

***Additive powers-of-two quantization:
an efficient non-uniform discretization
for neural networks***
Y. Li et al. (ICLR 2020)

Quantization (after clipping) can be uniform (write the number with k bits), power-of-two (less precision for larger numbers), or sum of powers of two, $q = 2^x + 2^y + 2^z$ (with k terms).

Learned step size quantization
S.K. Esser et al. (ICLR 2020)

To learn layer-specific quantization, with gradient descent, use the floor function in the forward pass, and relax it to the identity in the backward pass (using a different function in the forward and backward pass, to deal with locally constant functions, is a common trick). Rescale the step size with the size of each bin.

***Mixed precision DNNs:
all you need is a good parametrization***
S. Uhlich et al. (ICLR 2020)

In mixed precision DNN, the bitwidth is different for each layer, for the weights and the activations. For the uniform and pow-2 quantization schemes, do not directly estimate the bitwidth: parametrize the quantization with the maximum and the stepsize (the gradients are better-behaved). To train the model, add a penalty for the total memory required for the weights and activations; for the backward pass, set the derivative of the floor function to 1.

***Precision gating:
improving neural network efficiency
with dynamic dual-precision activations***
Y. Zhang et al. (ICLR 2020)

Dynamic quantization computes most features in low precision and a few (input-dependent) important ones in high precision, using a gating mechanism (precision gating).

***BinaryDuo: reducing gradient mismatch
in binary activation network
by coupling binary activations***
H. Kim et al. (ICLR 2020)

Binary neural nets, *i.e.*, neural nets whose weights and activations are ± 1 , are difficult to train. Approaches

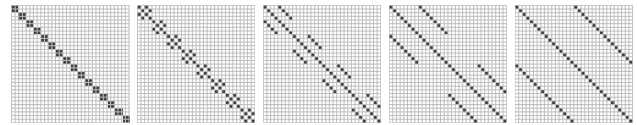
include coupling them with a higher-precision network, or using a surrogate, such as a stochastic binary neural net. To reduce the gradient mismatch problem: train coupled binary and ternary models; then decouple them and fine-tune the binary one.

***Critical initialisation in continuous
approximations of binary neural networks***
G. Stamatescu et al. (ICLR 2020)

Train a continuous surrogate for the binary neural net: a stochastic binary neural net.

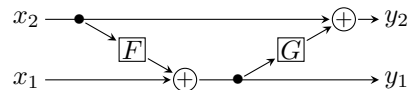
***Kaleidoscope: an efficient, learnable
representation for all structured linear maps***
T. Dao et al. (ICLR 2020)

To reduce the number of parameters in a neural net, one can use structured matrices instead of arbitrary, dense matrices: low-rank, sparse, Kronecker-factored, DFT, etc. Those matrices are products of sparse matrices, and they can be obtained from **butterfly matrices** (block matrices, whose blocks are diagonal); the construction encodes a recursive divide-and-conquer structure. The sparsity pattern is fixed, and learnable with gradient descent.



Reformer: the efficient transformer
N. Kitaev et al. (ICLR 2020)

The reformer is a memory-efficient variant of the transformer, using **reversible residual** layers (there is no longer any need to keep all the activations in memory), and locality-sensitive hashing (LSH) to locate and compute the largest elements of $\text{softmax}(Q'K)$ (which is almost sparse).



***Lite transformer
with long-short range attention***
Z. Wu et al. (ICLR 2020)

The attention matrix can be decomposed into local attention (dense, close to the diagonal) and global attention (sparse, off-diagonal): one can use a lower-dimensional attention mechanism for the global part, and a CNN for the local part.

***ELECTRA: pre-training text encoders
as discriminators rather than generators***
K. Clark (ICLR 2020)

Instead of masked pretraining, pretrain with replaced token detection. The replacements come from a small BERT, trained at the same time. (The model only requires 4 GPU-day.)

***ALBERT: a lite BERT for self-supervised
learning of language representations***
Z. Lan et al. (ICLR 2020)

Albert is a smaller BERT-like model, with a projection between the 1-hot word encodings and their vector embeddings, to reduce the number of parameters, shared parameters across layers, a sentence-order prediction loss, no dropout (there is enough data: it is not needed), and more data.

***FSPool: learning set representations
with featurewise sort pooling***
Y. Zhang et al. (ICLR 2020)

Transformer-based language models present visible patterns (vertical or horizontal lines, rectangles) in their self-attention heatmaps, which can be used to construct parse trees.

***Poly-encoders: transformer architectures
and pre-training strategies
for fast and accurate multi-sentence scoring***
S. Humeau et al. (ICLR 2020)

To decide if two sentences follow one another, the *bi-encoder* feeds each to a transformer and uses the scalar product of the outputs as score. The *cross-encoder* (BERT) feeds both sentences to a transformer (and can use cross-sentence attention) and directly outputs a score. The bi-encoder is less accurate, but much faster. The *poly-encoder* feeds the candidate sentence to a transformer that outputs a query, feeds the context to a transformer that outputs a sequence of states, and uses an attention mechanism to combine query and states.

***Are transformers universal approximators
of sequence-to-sequence functions?***
C. Yun et al. (ICLR 2020)

Transformers are universal approximators of continuous, permutation-equivariant sequence-to-sequence functions. Transformers with positional encoding are universal approximators of continuous sequence-to-sequence functions.

***Tree-structured attention
with hierarchical accumulation***
X.P. Nguyen et al. (ICLR 2020)

The attention mechanism can be generalized to hierarchical structures, e.g., parsed sentences.

***On the relationship between
self-attention and convolutional layers***
J.B. Cordonnier et al. (ICLR 2020)

Transformers (self-attention with positional encoding) also work for images: they can compute convolutions (you need as many heads as there are pixels in the receptive field).

***Multi-scale representation learning for spatial
feature distributions using grid cells***
G. Mai et al. (ICLR 2020)

The (1-dimensional) positional encoding used in transformers can be generalized to 2 dimensions, and it can be learned: Space2Vec is an embedding $(x, y) \mapsto$ latent representation.

***Neural text generation
with unlikelihood training***
S. Welleck et al. (ICLR 2020)

Standard language models produce too many frequent tokens and too few rare tokens. This may be due to maximum likelihood estimation: add an *unlikelihood* penalty to explicitly penalize negative tokens, defined as tokens previously used in the sentence.

The curious case of neural text degeneration
A. Holtzman et al. (ICLR 2020)

Text-generation models (e.g., GPT2), relying on beam search, generate text less surprising than humans do. Instead of beam search, which is an approximation of maximum likelihood, use sampling, but discard the most unlikely words, not by retaining the top- k proposals, but by retaining the top- p (probability mass) ones ($p = 0.95$).

***Towards verified robustness
under text deletion interventions***
J. Welbl et al. (ICLR 2020)

Language models suffer from undersensitivity: deleting parts of a sentence can make a model more confident, because it exploits shallow clues (negation, premise-only entailment, etc.). To avoid those problems, ensure that removing words does not increase the probability, e.g., with interval bound propagation (IBP).

***BERTScore:
evaluating text generation with BERT***
T. Zhang et al. (ICLR 2020)

Text generation evaluation metrics are often based on n -grams (BLEU, ROUGE, METEOR, chrF), but they struggle with synonyms. More recent metrics (Meant, YiSi, BERTScore) are based on embeddings. BERTScore computes the BERT embeddings of the two sentences (generated and reference), then all the cosine similarities; it then greedily matches the words (in each directions, to separate precision from recall) averages the similarities, and computes an F_1 score.

***Improving neural language generation
with spectrum control***
L. Wang et al. (ICLR 2020)

The word embeddings of current language models do not span the whole available space; this can be seen in the distribution of the singular values. Instead of learning an embedding $x \mapsto Wx$ from 1-hot-encoded vectors, learn its SVD decomposition $W = U\Sigma V'$, with penalties to enforce the orthogonality of U and V and to make the k th singular value close to $ck^{-\gamma}$ or $c_1 \exp(-c_2 k^\gamma)$

Mirror-generative neural machine translation
Z. Zheng et al. (ICLR 2020)

Non-parallel (*i.e.*, monolingual) data can help train better NMT systems, *e.g.*, using back-translation (translated sentences should sound idiomatic). The 4 models (source and target language models, and the translation models in both directions) can be linked with a VAE whose latent variable models the contents of the sentence, trained iteratively (one direction at a time).

Logic and the 2-simplicial transformer
J. Clift et al. (ICLR 2020)

Attention can be generalized to higher-order relations (between 3 objects instead of 2): use two keys instead of one, and replace the scalar product of query and key with the tetrahedron volume

$$\langle a, b, c \rangle = (a \cdot b)c - (a \cdot c)b + (b \cdot c)a.$$

***Strategies for pre-training
graph neural networks***
W. Hu et al. (ICLR 2020)

To limit negative transfer in graph neural nets, pre-train both node embedding (predict (masked) node labels, or context) and graph embeddings (various supervised tasks: attribute prediction, structural similarity).

***Composition-based multi-relational
graph convolutional networks***
S. Vashishth et al. (ICLR 2020)

To process knowledge graphs with GNN, learn both node and edge embeddings.

***InfoGraph: unsupervised and semi-supervised
graph-level representation learning
via mutual information maximization***
F.Y. Sun et al. (ICLR 2020)

There are two common representations of graphs:

- Graph kernels decompose graphs into substructures/motifs and counts them;
- Graph2vec compute node embeddings, and aggregates them (*cf.* word2vec, doc2vec).

InfoGraph uses graph convolutions, and a discriminator, fed with pairs of graph and subgraph representations, to decide if they are from the same graph. For semi-supervised learning, use two encoders, one with the supervised loss, one with the unsupervised one, and keep them close to one another by maximizing the mutual information between their activations – this can be done by a discriminator, fed with a representation from each encoder, and deciding if they correspond to the same graph (*cf.* student/teacher networks).

***What graph neural networks cannot learn:
depth vs width***
A. Loukas (ICLR 2020)

Message-passing GNNs are not universal, unless nodes have uniquely identifying features, and $\text{depth} \times \text{width} = \Omega(n^2)$ (the exponent is lower for some tasks).

PairNorm: tackling oversmoothing in GNNs
L. Zhao and L. Akoglu (ICLR 2020)

GNNs achieve best performance with 1 or 2 layers – adding more layers leads to vanishing gradients, overfitting, and “oversmoothing”.

***Learning deep graph matching
with channel-independent embedding
and Hungarian attention***
T. Yu et al. (ICLR 2020)

The attention mechanism can also be generalized to graph neural nets (GNN).

***Learning to retrieve reasoning paths over
wikipedia graph for question answering***
A. Asai et al. (ICLR 2020)

When searching for the answer of a question in a text database (*e.g.*, Wikipedia), the paragraph containing the answer may not have a lot of overlap (often measured at the word level, *e.g.*, tf-idf): multi-hop reasoning may be needed. Use an RNN to learn the path in the graph.

***Differentiable reasoning
over a virtual knowledge base***
B. Dhingra et al. (ICLR 2020)

Question answering over text often requires combining information from several sources. The traditional approach (retrieve the relevant documents, understand them, answer) struggles with such multi-hop questions: we do not know which documents to retrieve until we have partially answered the question.

Encode entities as vectors and relations as sparse matrices; use BERT to measure the similarity between relations and mentions; information retrieval can then be done with sparse matrices products and approximate k -NN queries.

***You can teach an old dog new tricks!
On training knowledge graph embeddings***
D. Ruffinelli et al. (ICLR 2020)

Implementation details (knowledge graph embedding) matter: small tricks can make old models (whose training did not use them) look as good as recent models (whose training did).

***Graph constrained reinforcement learning
for natural language action spaces***
P. Ammanabrolu and M. Hausknecht

Reinforcement learning algorithms struggle with the discrete but very large action space of text adventure games such as Zork. Use a knowledge graph representing (what we know of) the game world, with a graph attention network (GAT), to trim down the search space (Verb Object1 Object2).

***Query2box: reasoning over knowledge graphs
in vector space using box embeddings***
H. Ren et al. (ICLR 2020)

To answer first-order logic queries on a knowledge graph, model entities as points (in latent space), and queries as boxes (or unions of boxes); relations map entities to boxes, e.g., (S,V,?) or (?,V,O).

***Tensor decompositions
for temporal knowledge base completion***
T. Lacroix et al. (ICLR 2020)

The ComplEx model

$$X_{i,j,k} = \langle U_i, V_j, \bar{U}_k \rangle$$

is a low-rank tensor completion model, for knowledge bases, often used with a nuclear 3-norm regularizer. It can be generalized to temporal knowledge bases,

$$X_{i,j,k,t} = \langle U_i, V_j, \bar{U}_k, T_t \rangle$$

or (temporal and non-temporal predicates)

$$X_{i,j,k,t} = \langle U_i, \tilde{V}_j + V_j \odot T_t, \bar{U}_k \rangle,$$

with a temporal regularizer

$$\sum \|T_{t+1} - T_t\|_p^p.$$

***LambdaNet: probabilistic type inference
using graph neural networks***
J. Wei et al. (ICLR 2020)

Computer code (untyped Python, TypeScript) can be parsed into a tree (an abstract syntax tree, AST), which can be fed to a graph neural net and a pointer network, for various prediction tasks: type inference, bug detection, etc.

The compiler can provide more links to add to the graph.

***Hoppity: learning graph transformations
to detect and fix bugs in programs***
E. Dinella et al. (ICLR 2020)

A (Javascript) program can be represented as an abstract syntax tree (AST), which a graph neural net can turn into a latent representations. A bugfix is a graph edit, represented by a sequence of operations.

***Towards a deep network architecture
for structured smoothness***
H. Habeeb and O. Koyejo (ICLR 2020)

CNNs gather information from fixed neighbourhoods: instead, the “fixed grouping layer” (FGL) uses neighbourhoods defined by data, e.g., with some clustering algorithm (fMRI data) or external data (industries, in finance), and ensures that the receptive fields do not span group boundaries.

***Deep 3D pan
via adaptive t-shaped convolutions
with global and local adaptive dilations***
J.L.G. Bello and M. Kim (ICLR 2020)

The filters used by convolutions need not be square: for some applications, T-shaped ones make sense.

***Denoising and regularization via exploiting
the structural bias of convolutional generators***
R. Heckel and M. Soltanolkotabi (ICLR 2020)

An auto-encoder (U-Net) trained on a single image ($n = 1$) first learns the image, then the noise – early stopping provides denoising. This comes from the linear upsampling, which fits the smooth part (low frequency) before the noise.

Adjustable real-time style transfer
M. Babaeizadeh and G. Ghiasi (ICLR 2020)

***Robust and interpretable blind image denoising
via bias-free convolutional neural networks***
S. Mohan et al. (ICLR 2020)

Bias-free CNNs are better at denoising.

***Neural symbolic reader: scalable integration
of distributed and symbolic representations
for reading comprehension***
X. Chen et al. (ICLR 2020)

Language models (BERT) cannot easily answer simple questions requiring arithmetic computations. Adding specialized modules (addition, subtraction, count, span, negation, etc.) does not allow compositionality and does not generalize well to new problems. The neural semantic reader extracts the structure of the text and the question, turns the question into a program, and runs it on the parsed text.

Neural arithmetic units

A. Madsen and A.R. Johansen (ICLR 2020)

Neural networks are good at interpolation, but not at extrapolation: arithmetic expressions perform better. For instance, the neural arithmetic logic unit (NALU) is defined as

$$\begin{aligned}w &= \tanh \times \text{sigmoid} && (\text{sign and scale}) \\a &= \sum w_i x_i && (\text{sum}) \\b &= \exp \sum w_i \log(|x_i| + \varepsilon) && (\text{product}) \\g &= \text{sigmoid} && (\text{gate}) \\y &= ga + (1 - g)b && (\text{output})\end{aligned}$$

But it does not consistently find the correct solution (and negative numbers are a problem). Instead, the neural arithmetic unit, biases w towards -1, 0 and +1, and uses an actual multiplication, with a gating mechanism to select what to multiply.

What can neural networks reason about? K. Xu et al. (ICLR 2020)

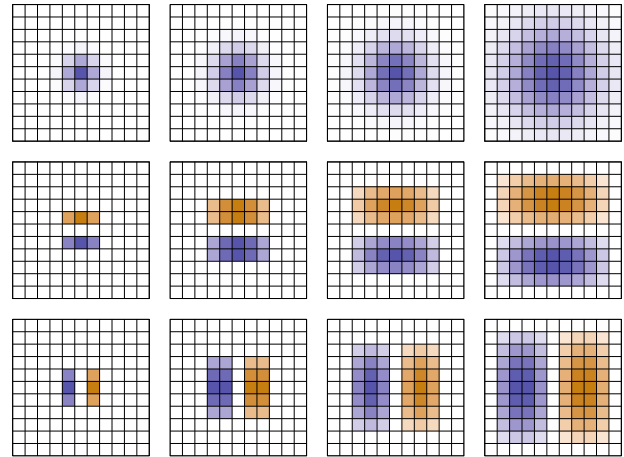
A given neural network architecture (e.g., GNN) has an implicit prior on the structure of the computations/algorithm it learns: it will more easily learn algorithms aligned with that structure. The GNN nested loops (loop over the nodes, loop over the neighbours, apply some function) are aligned with dynamic programming algorithms: this explains their success. They do not work for NP-complete problems, though: try “neural exhaustive search” (NES) instead.

Learn to explain efficiently via neural logic inductive learning Y. Yang and L. Song (ICLR 2020)

Neural networks can be used to explain an image by mapping its elements (e.g., car, wheel, window, etc.) to vectors, and relations to matrices, and using an attention mechanism to find first-order-logic formulas matching the scene.

Scale-equivariant steerable networks I. Sosnovik et al. (ICLR 2020)

CNNs are translation-equivariant, but not scale-equivariant. Use a predefined basis of filters, at different scales, and learn a (scale- and rotation-independent) linear combination of them.



B-spline CNNs on Lie groups E.J. Bekkers (ICLR 2020)

DeepSphere: a graph-based spherical CNN M. Defferrard et al. (ICLR 2020)

Co-attentive equivariant neural networks: focusing equivariance on transformations co-occurring in data D.W. Romero and M. Hoogendoorn

Group equivariant CNNs apply the same filter after the various actions of a (discrete) group. Add an attention mechanism to identify co-occurring actions.

On universal equivariant set networks N. Segol and Y. Lipman (ICLR 2020)

\mathfrak{S}_n -equivariant affine maps are of the form $X \mapsto XA + \mathbf{1}\mathbf{1}'XB + \mathbf{1}c'$ (DeepSets). If $B = 0$, each row of the input is transformed in the same way (PointNet).

Pure and spurious critical points: a geometric study of linear networks M. Trager et al. (ICLR 2020)

The geometry of the weight space differs from that of the function space: one can distinguish between critical points in function space (“pure”) and those only in parameter space (“spurious”). For linear networks and quadratic loss, spurious critical points are saddle points and there are no bad minima: all local minima are global, even if the network has a bottleneck layer – in this case, the map is not full rank, and the space of such maps, the *determinantal variety*, is non-convex and non-smooth.

Efficient Riemannian optimization on the Stiefel manifold via the Cayley transform J. Li et al. (ICLR 2020)

If a matrix W is skew symmetric, then $(I + W)(I - W)^{-1}$ is orthogonal (Cayley transform). The paths

$Y(\alpha) = (1 - \frac{1}{2}\alpha W)^{-1}(1 + \frac{1}{2}\alpha W)X$, or their iterative approximation $Y(\alpha) = X + \frac{1}{2}\alpha W(X + Y(\alpha))$, allow for Riemannian optimization on the Stiefel manifold (the space of orthogonal matrices) – orthogonal weight matrices improve learning, accuracy and robustness.

On the need for topology-aware generative models for manifold-based defenses
U. Jang et al. (ICLR 2020)

Generative models assume that the distribution in the latent space is a standard Gaussian – in particular, the data manifold is contractible. In reality, it could be more complex – in particular, there could be several connected components.

AE-OT: a new generative model based on extended semi-discrete optimal transport
D. An et al. (ICLR 2020)

The optimal transport map from a uniform distribution to a multi-modal one can be discontinuous and can lead to mode collapse. It is the gradient of the Brenier potential, which is continuous and convex: directly learn the Brenier potential (in a latent representation),

Deep orientation uncertainty learning based on a Bingham loss
I. Gilitschenski et al. (ICLR 2020)

The Bingham distribution, on the hypersphere, is

$$p(x) \propto \exp(x' M Z M' z)$$

where $x \in \mathbf{S}^{d-1}$, $M \in O(d)$ and $Z \in \mathbf{R}^{d \times d}$ is diagonal. To estimate the orientation of an object, as a unit quaternion, have a neural network estimate M (with Gram-Schmidt orthonormalization) and Z , and use the log-likelihood as loss function. The normalizing constant is difficult to compute: use a pre-computed look-up table and a radial basis function (RBF) interpolator.

Neural Tangents: fast and easy infinite neural networks in Python
R. Novak et al. (ICLR 2020)

Neural-Tangents provide infinitely-wide layers, as a drop-in replacement for `stax`, the `jax` neural network library. Since these are Gaussian processes, they do not scale well with dataset size.

Harnessing the power of infinitely wide deep nets on small-data tasks
S. Arora et al. (ICLR 2020)

The neural tangent kernel (NTK) works better than random forests on the UCI dataset, and better than ResNets on small datasets (10 to 100 samples).

DDSP: differentiable digital signal processing
J. Engel et al. (ICLR 2020)

DDSP provides digital signal processing operations (oscillators, filters, etc. – everything you would do in an audio editor such as Audacity or with a music programming language such as ChuckK), as TensorFlow layers (non-reproducible) examples include removing the reverb (or extracting it to add it to another recording) or changing the timbre (violin→flute, human→violin).

BackPACK: packing more into backprop
F. Dangel et al. (ICLR 2020)

BackPACK is a PyTorch library to compute more quantities during back-propagation:

- the individual gradients from a mini-batch (for instance to identify which samples are informative and which are not, for importance sampling),
- the variance of the gradients in a mini-batch (to monitor the signal-to-noise ratio and increase the batch size if needed),
- a Fisher information matrix approximation.

Also check: `PyHessian`, `JAX` (or `Zygote`, in Julia) (both forward and backward AD, AD through arbitrary programs: loops, conditionals).

DiffTaichi: differentiable programming for physical simulation
Y. Hu et al. (ICLR 2020)

At stability's edge: how to adjust hyperparameters to preserve minima selection in asynchronous training of neural networks?
N. Giladi et al. (ICLR 2020)

To ensure stability in (centralized) asynchronous SGD, rescale the learning rate: learning rate $\propto 1/\text{delay}$.

Gap aware mitigation of gradient staleness
S. Barkai et al. (ICLR 2020)

Distributed stochastic gradient descent can rely on:

- Ignoring synchronization problems;
- Waiting until all the workers have finished;
- Staleness awareness: divide the learning rate by the delay: $\eta \mapsto \eta/(\tau + 1)$, but this over-penalizes old gradients;
- Gap-awareness: use the gap (minimum number of updates needed to move from the previous to the current parameters):

$$1 + \frac{\|\theta_t - \theta_{t-\tau}\|}{\eta \cdot \mathbb{E}[\nabla \text{loss}]}$$

Decentralized deep learning with arbitrary communication compression
A. Koloskova et al. (ICLR 2020)

ChocoSGD combines decentralized SGD with (lossy) communication compression: do not send everything,

do not send accurate numbers, but keep track of the cumulated errors, to send them later, once they have accumulated.

SQIL: imitation learning via reinforcement learning with sparse rewards
S. Reddy et al. (ICLR 2020)

Imitation learning suffers from distribution shift: if the agent ends up away from the trajectory, it no longer knows what to do. Adversarial approaches (e.g., GAIL) use a discriminator, separating expert from agent trajectories, but GAN training is unstable. Instead of learning a reward function, set the reward of the expert actions to 1, those of the agent to 0, put both in the replay buffer, and use off-policy learning (e.g., soft actor critic).

Optimistic exploration even with a pessimistic initialisation
T. Rashid et al. (ICLR 2020)

Optimistic initialization does not work well with function approximation. Use optimistic Q-values:

$$Q^+(s, a) = Q(s, a) + C/(N(s, a) + 1)^M.$$

Dynamical distance learning for semi-supervised and unsupervised skill discovery
K. Hartikainen et al. (ICLR 2020)

RL problems often have sparse rewards: the agent only receives a reward when it reaches the goal. Learn a distance function, measuring how many time steps away two states are, on trajectories for the current policy. Then, use minus this distance as a reward, to improve the policy.

This generalizes the discounted state value function, which can be seen as a measure of the distance to the final goal – instead of the distance between arbitrary states.

Imitation learning via off-policy distribution matching
I. Kostrikov et al. (ICLR 2020)

ValueDICE is an alternative to DAC, GAIL, based on the *Donsker-Varadhan* representation of the KL divergence

$$\text{KL}(p||q) = \max_{f: X \rightarrow \mathbf{R}} \mathbb{E}_{x \sim p} [f(x)] - \log \mathbb{E}_{x \sim q} [e^{f(x)}].$$

Implementation matters in deep RL: a case study on PPO and TRPO
L. Engstrom et al. (ICLR 2020)

Apparently insignificant code-level optimization (value-clipping, reward-clipping, Adam annealing, orthogonal initialization, etc.) have a large impact – more than the choice of algorithm (e.g., PPO vs TRPO).

Explain your move: understanding agent actions using specific and relevant feature attribution
N. Puri et al. (ICLR 2020)

To explain the action chosen by an agent, perturb the state features and re-compute the Q-value: if it changes a lot (in absolute value, and/or relatively to the Q-values of the other actions), the feature was important.

Disagreement-regularized imitation learning
K. Brantley et al. (ICLR 2020)

To limit the compounding error problem in behavioural cloning, add a penalty term to stay close to the expert distribution.

RIDE: rewarding impact-driven exploration for procedurally-generated environments
R. Raileanu and T. Rocktäschel (ICLR 2020)

Reward actions that have a large impact on the environment, measured by the L^2 norm of the difference between the latent state representations. To focus on unforeseen/surprising consequences, train a model to predict the next (latent) state from the current state and the action. To focus on changes we can actually control, train a model to predict the action from the current and next state.

Toward evaluating robustness of deep reinforcement learning with continuous control
T.W. Weng et al. (ICLR 2020)

Threat models in reinforcement learning include:

- Observation manipulation (before the agent receives it);
- Action manipulation (after the agent chooses it).

To prevent those threats, one could learn the dynamics, identify unsafe states, and avoid them – but performance degrades a lot.

Keep doing what worked: behavior modelling priors for offline reinforcement learning
N. Siegel et al. (ICLR 2020)

When learning from a fixed replay buffer, with no new trajectories, avoid actions that would lead away from the explored portions of the state-action space, by putting a prior on the policy, to make it close to those in the buffer – or, better, close to the actions that worked.

Model-based reinforcement learning for biological sequence design
C. Angermueller et al. (ICLR 2020)

To find biological sequences x maximizing $f(x)$ (the result of some wet-lab experiment result), use reinforcement learning, where the actions select the next character of the (fixed-length) sequence, and we only receive

a reward, $f(x)$, at the end. Add a penalty to increase diversity.

Harnessing structures for value-based planning and reinforcement learning
Y. Yang et al. (ICLR 2020)

Represent the Q -value as a matrix, state \times action (after discretization, if needed): quite often, it is low-rank. Use such a low-rank matrix in planning or reinforcement learning.

The ingredients of real-world robotic reinforcement learning
H. Zhu et al. (ICLR 2020)

Real-world reinforcement learning faces three problems:

- No resets (the robot could get stuck – or could reach the goal, from which the optimal action is to do nothing): learn a perturbation agent;
- High-dimensional observations (images) instead of low-dimensional state: learn a low-dimensional representation of the state, with a VAE;
- No reward: train a classifier to recognize the goal (from a set of example images).

Behaviour suite for reinforcement learning
I. Osband et al. (ICLR 2020)

bsuite provides a set of experiments and metrics (exploration, generalization, credit assignment, scalability, etc.) to test and compare RL algorithms (like MNIST for RL). It can also generate a 1-page L^AT_EXsummary, to add to your paper.

Evolutionary population curriculum for scaling multi-agent reinforcement learning
Q. Long et al. (ICLR 2020)

In multi-agent reinforcement learning, start with small populations, and progressively increase the size. The model, for each agent, should be applicable to an arbitrary number of inputs (number of enemies, resources). When increasing the population size, we do not only want agents that perform well when the population size is N : they should also be good initializations for population size $2N$ – use an evolutionary strategy, with k groups of agents, and keep those performing best at size $2N$. Application: grassland game (grass, sheep, wolves).

Demystifying inter-class disentanglement
A. Gabbay and Y. Hoshen (ICLR 2020)

To disentangle class (e.g., identity, in a face recognition system) from content (pose), learn separate class and content embeddings but, to avoid information leakage, have the class embedding be the same for all members of that class.

Learning representations for binary-classification without backpropagation
M. Lechner (ICLR 2020)

Backpropagation is biologically implausible. Alternatives include: Hebbian learning, Target propagation, Feedback alignment (FA), direct feedback alignment (FDA), monotone DFA.

Reconstructing continuous distributions of 3D protein structure from Cryo-EM images
E.D. Zhong (ICLR 2020)

Each Cryo-EM (electron microscopy) image contains only one molecule, in a random orientation and location, but they may not all be in the same configuration. We usually try to cluster them. Instead, assume the configurations form a 2-dimensional manifold. For the orientation and location of the molecule in each image, use an optimization over $SO(3) \times \mathbb{R}^2$

Intrinsically motivated discovery of diverse patterns in self-organizing systems
C. Reinke et al. (ICLR 2020)

To find interesting and diverse patterns in a self-organizing system (snowflakes, animal skin patterns, or, as in this example, the continuous game of life – time and space are still discrete, but the values are in $[0,1]$), start with a random initial state and look after 200 steps.

To ensure diversity in the output, learn a latent representation (8-dimensional VAE) of the outputs; pick a point at random in this latent space; find an input whose output is close to that point; iterate (and retrain the VAE once in a while).

Since random noise tends to generate global patterns rather than local ones, use patterns from a CPPN (compositional pattern-producing network) as inputs.

Learning compositional Koopman operators for model-based control
Y. Li et al. (ICLR 2020)

Koopman operator theory takes a non-linear dynamical system $x_{n+1} = F(x_n)$ and finds an embedding into a higher-dimensional space $y = g(x)$ such that the dynamics become linear $y_{n+1} = Ky_n$. Systems made of many parts are often modeled (and controlled) with interaction graphs. Those two approaches can be combined, the graph structure corresponding to a block Koopman matrix.

SNODE: spectral discretization of neural ODEs for system identification
A. Quaglini et al. (ICLR 2020)

Separate the neural ODE “find θ so that x defined by $\dot{x}(t) = f_\theta(t, x(t))$, $x(t_0) = y_0$ minimizes $\int \text{Loss}(t, x(t))dt$ ”, into two:

- Find a function x minimizing the loss;

– Find θ so that this x approximately solves the ODE.
Estimate the derivative \dot{x} by expressing x with Legendre polynomials. This is much faster than using an ODE solver.

***On robustness
of neural ordinary differential equations***
H. Yan et al. (ICLR 2020)

Neural ODEs are more robust to noise than CNNs with residual connections

$$\begin{aligned}\dot{z}(t) &= f_\theta(z(t), t) \\ z(0) &= z_{\text{in}} \\ z(1) &= z_{\text{out}}\end{aligned}$$

For time-invariant steady neural ODE (TisODE)

$$\dot{z}(t) = f_\theta(z(t))$$

one can add a further robustness penalty.

***Deep audio priors emerge from
harmonic convolutional networks***
Z. Zhang et al. (ICLR 2020)

Auto-encoders (U-Net) can denoise images, but not sound: replace the convolutions with **harmonic convolutions** (a form of dilated convolutions): do not use $[(1 - \varepsilon)f, (1 + \varepsilon)f]$ as neighbourhood of frequency f , but $\{f, 2f, 3f, \dots, 12f\}$.

***On the steerability
of generative adversarial networks***
A. Jahanian et al. (ICLR 2020)

One can learn transformations (rotation, translation, zoom, colour changes, etc.) in the latent space of GANs, to some extent (not beyond the training distribution).

***Controlling generative models
with continuous factors of variations***
A. Plumerault et al. (ICLR 2020)

Yet another attempt to control the output of a GAN: size, orientation, colour, etc.

Real or not real, that is the question
Y. Xiangli et al. (ICLR 2020)

The Realness GAN considers that samples are not fully real or fully fake, and optimizes

$$\max_G \min_D \mathbb{E}_{x \sim \text{data}} \text{KL}(A_1 \| D(x)) + \mathbb{E}_{x \sim \text{fake}} \text{KL}(A_0 \| D(x))$$

where A_0 and A_1 are random variables (e.g., $A_0 \sim \text{Beta}(5, 1)$, $A_1 \sim \text{Beta}(1, 5)$).

***Dynamic time lag regression:
predicting what & when***
M. Chandorkar et al. (ICLR 2020)

The dynamic time-lag regression $y(t + g(x_t)) = f(x_t)$ (with f and g unknown) has a closed-form log-likelihood. It has applications in astronomy (solar wind prediction).

***Intensity-free learning
of temporal point processes***
O. Shchur et al. (ICLR 2020)

When modeling temporal point processes (TPP), instead of using an RNN to compute the estimate the intensity function λ_t , estimate the density function of the time until the next event, with normalizing flows (to transform a Gaussian density into the desired density) or a Gaussian mixture.

***Variational autoencoders for highly
multivariate spatial point processes intensities***
B. Yuan et al. (ICLR 2020)

Instead of modeling the intensity function λ , model the density $h(x) = \lambda(x) / \int \lambda$ (this is how one can sample from the process: first sample the number of points N , from a Poisson distribution, then sample N points, iid, from h), with a VAE.

***On solving minimax optimization locally:
a follow-the-ridge approach***
Y. Wang et al. (ICLR 2020)

The nonconvex-nonconcave minimax problem

$$\min_x \max_y f(x, y)$$

can be solved with the follow-the-ridge algorithm, which requires second-order information (the gradient is zero along the ridge).

$$\begin{aligned}x_{t+1} &\leftarrow x_t - \eta \nabla_x f(x_t, y_t) \\ y_{t+1} &\leftarrow y_t - \eta \nabla_y f(x_t, y_t) + \eta H_{yy}^{-1} H_{yx} \nabla_x f(x_t, y_t)\end{aligned}$$

***Gradientless descent:
high-dimensional zeroth-order optimization***
D. Golovin et al. (ICLR 2020)

When the gradient of the function to minimize is not available, gradient-less methods (moving in a random direction if it improves the objective) outperform gradient-approximating ones such as accelerated random search (ARS),

$$x \leftarrow x - \varepsilon \frac{f(x + hv) - f(x)}{h} v$$

for a random unit vector v .

***On mutual information maximization
for representation learning***
M. Tschannen et al. (ICLR 2020)

When learning a latent representation, we want the latent representation to be simpler (lower-dimensional) but to keep as much information as possible from the input. This can be achieved by maximizing the mutual information $MI(x, g(x))$ between the input x and the latent representation $g(x)$ (InfoMax). Modern variants maximize $MI(g(x_1), g(x_2))$, where x_1 and x_2 are different parts of the same image (e.g., top and bottom). The mutual information is maximum if g is bijective, and is invariant by bijections – it is unlikely to find good representations. However, the approximations of MI used (NCE, noise contrastive estimation, or NWJ) are not invariant under bijections, and prefer (hard-to-invert maps that are) good representations.

***Understanding the limitations of variational
mutual information estimators***
J. Song and S. Ermon (ICLR 2020)

***Learning robust representations
via multi-view information bottleneck***
M. Federici et al. (ICLR 2020)

If we know that two images correspond to the same label (but do not know the label), we can still learn a latent representation by maximizing the mutual information between the latent representations of the two images (to keep what is common and discard what is not) and minimizing the divergence between the distributions of those latent representations (in the case of a VAE). For isolated images, use data augmentation.

Rényi fair inference
S. Baharlouei et al. (ICLR 2020)

Try the Rényi correlation

$$\rho(X, Y) = \sup_{\substack{f, g \\ E f(X) = E g(Y) = 0 \\ E f(X)^2 = E g(Y)^2 = 1}} [f(X)g(Y)]$$

instead of mutual information (for discrete variables, it is tractable, especially when one is binary). To make a model fairer (classifier or clustering), add a penalty for the Rényi correlation between the output and the protected attribute.

Overlearning reveals sensitive attributes
C. Song and V. Shmatikov (ICLR 2020)

Sensitive attributes can be censored using adversarial training (it should be impossible to predict the sensitive attribute from the latent representation) or information theory (by minimizing the mutual information between the latent representation and the sensitive attribute). Unfortunately, such censoring is either inefficient or harmful to the model.

***Training individually fair ML models
with sensitive subspace robustness***
M. Yurochkin et al. (ICLR 2020)

A model is *individually fair* if it is robust to sensitive perturbations.

***Understanding why neural networks
generalize well through GSNR of parameters***
J. Liu et al. (ICLR 2020)

The one-step generalization ratio is the ratio of the change in test loss over the change in training loss.

The gradient signal-to-noise ratio is the ratio of the squared norm of the mean gradient over the variance of the gradients computed on individual samples; it measures if the gradients of the different samples agree.

If the gradients are consistent across samples, the GSNR and OSGR are large, and the model generalizes well.

***Stable rank normalization for improved
generalization in neural networks and GANs***
A. Sanyal et al. (ICLR 2020)

The generalization gap depends on (the product of) the spectral norms of the weight matrices (Lipschitzness) and (the sum of) their stable ranks.

The *stable rank* of a matrix W with singular values σ_i 's is

$$\text{srank}(W) = \frac{\|W\|_F^2}{\|W\|_2^2} = \frac{\sum \sigma_i^2}{\sigma_1^2} \leq \text{rank}(W)$$

***Deep probabilistic subsampling
for task-adaptive compressed sensing***
I.A.M. Huijben et al. (ICLR 2020)

Gumbel max sampling

$$\underset{i}{\text{Argmax}} \log p_i + g_i, \text{ for } g_i \sim \text{Gumbel}(0, 1)$$

can be made differentiable by replacing the argmax with a softmax (with temperature), and trained to decide which Fourier frequencies to sample to best help classify or reconstruct an input image.

***A function space view of bounded norm
infinite width ReLU nets: the multivariate case***
G. Ongie et al. (ICLR 2020)

The r -norm measures the representation cost of a function

$$R(f) = \liminf_{\varepsilon \rightarrow 0} \inf_{\theta} \{\|\theta\|^2 : \|h_{\theta} - f\| \leq \varepsilon\}$$

where h_{θ} is a 1-hidden-layer infinite-width ReLU network. It can be characterized as $R(f) = c \cdot \|\mathcal{R}(-\Delta)^{(d+1)/2} f\|_1$ where Δ is the Laplacian and \mathcal{R} the Radon transform. In dimension 1: $R(f) = \int |f''|$. It is not a RKHS norm (as with the NTK). Some functions have infinite $R(f)$, even though they are well approximated with 2-hidden-layer networks.

***Knowledge consistency
between neural networks and beyond***
R. Liang et al. (ICLR 2020)

Given two trained networks performing the same task, compare their latent representations x_A , x_B as follows:

- Try to express x_B as a linear transformation of x_A ;
- Try to express the residuals as a 1-layer non-linear transformation of x_A ;
- Try to express the residuals as a 2-layer non-linear transformation of x_A ;
- The residuals are then noise.

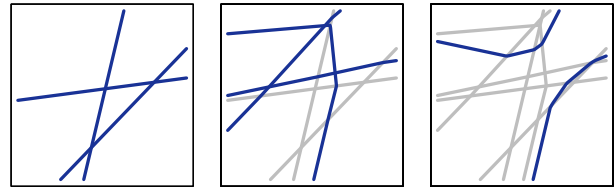
This gives a decomposition of the latent representation, which can help spot unreliable features, measure overfitting, and compress networks.

***Functional vs. parametric equivalence
of ReLU networks***

M. Phuong and C.H. Lampert (ICLR 2020)

The function computed by a (ReLU) network is invariant under: permutation of neurons, rescaling (of two layers, with inverse scaling factors). These are the only function-preserving transformations: the *fold-sets* of a ReLU network provide enough constraints on the

weights.



***Stochastic AUC maximization
with deep neural networks***
M. Liu et al. (ICLR 2020)

AUC maximization can be formulated as a min-max problem, non-convex and concave, solvable with the approximate proximal point method (APPM).

***Playing the lottery with rewards and multiple
languages: lottery tickets in RL and NLP***
H. Yu et al. (ICLR 2020)

The “lottery ticket hypothesis” is not limited to image classification: it also seems to hold in NLP and RL.

Deep equilibrium models S. Bai et al. (NeurIPS 2019)

A deep neural network with *weight-tied*, input-injected layers computes the repeated application of a function $z_{i+1} = f_{\theta}(z_i, x)$; it converges (in practice) to an equilibrium point $z^* = f_{\theta}(z^*, x)$. **Deep equilibrium (DEQ)** models find this equilibrium by root finding rather than iteration. They can replace deep neural networks (TrellisNet, universal transformer) with 1-layer networks: they are more memory-efficient.

List-decodable linear regression S. Karmalkar et al. (NeurIPS 2019)

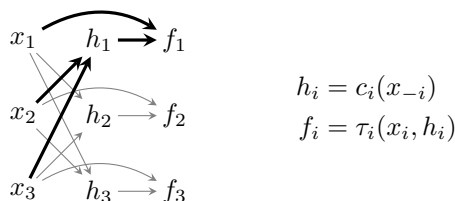
List-decodable models are robust methods returning, not one estimator, but a list of estimators, with a good one ($\|\hat{\alpha} - \alpha^*\| < \varepsilon$) with high probability. They allow for adversarial outliers, and contamination beyond 50%.

Legendre memory units: continuous-time representation in recurrent neural networks A. Voelker et al. (NeurIPS 2019)

Strided recurrent neural nets can account for long-term dependencies, with Legendre polynomials: they are an alternative to LSTMs. [They also do hardware, and have implemented their RNN as a spiking neural net.]

Neural networks with cheap differential operators R.T.Q. Chen and D. Duvenaud (2019)

The divergence $\text{div} \cdot f = \sum \partial f_i / \partial x_i$ of a neural net $f : \mathbf{R}^d \rightarrow \mathbf{R}^d$ only requires the diagonal of the Jacobian. It can be efficiently computed by *hollow networks*

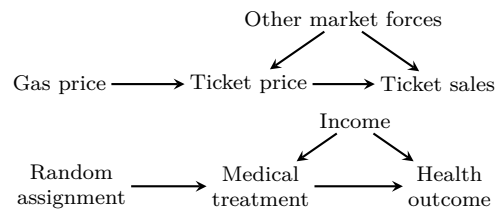
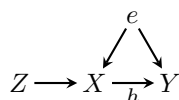


(the Jacobian can easily be separated into diagonal and non-diagonal parts, your deep learning framework can be asked to only compute the gradient of the diagonal).

Applications include fixed point iterations, normalizing flows.

Kernel instrumental variable regression R. Singh et al. (NeurIPS 2019)

To learn a causal relation in presence of confounding variables, regression is biased: it gives a prediction, not a *counterfactual prediction*.



Use an instrumental variable Z (which influences X but not directly Y) and 2SLS:

$$X \sim Z$$

$$Y \sim \hat{X}(Z) \quad \text{estimated on disjoint samples.}$$

Replace the linear regressions with kernel ridge regressions. Asymmetric sample splitting is important, with more observations in the first stage, and even more if the relation $Z \rightarrow X$ is less smooth. DeepIV is preferable if there is enough data (10,000 observations) and the relation is complex.

Adversarial samples are not bugs, they are features A. Ilyas et al. (NeurIPS 2019)

Take a dataset with cats and dogs, train a model, replace each sample with an adversarial example, and flip the labels accordingly, and learn a new model: it works well on the test set. The new model uses non-robust features that generalize well.

Quantum entropy scoring for fast robust mean estimation and improved outlier detection Y. Dang et al. (NeurIPS 2019)

PCA is a good way to find outliers, $\text{score}(x_i) = \langle x_i, \text{PCA}_k \rangle$, but it only detects outliers in one direction at a time: it is not ideal in high dimension. *Quantum entropy scoring* (QUE) replaces PCA with regularized PCA

$$W = \underset{\text{tr } U=1, U \succcurlyeq 0}{\text{Argmax}} \quad \alpha \langle U, \Sigma \rangle + \text{tr}(U \log U)$$

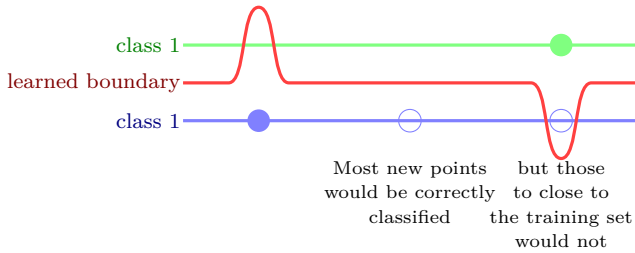
$$\text{score}(X) = \langle XX', W \rangle$$

Uniform convergence may be unable to explain generalization in deep learning V. Nagarjan and J.Z. Kotler (NeurIPS 2019)

To explain why overparametrized models generalize well, we can use uniform convergence to find bounds of the form

$$\text{generalization gap} \leq O \left(\frac{\text{effective model size}}{\text{training set size}} \right)$$

where the effective model size accounts for the bias of the training algorithm (SGD). Those bounds are too loose or require unreasonable assumptions. They do not even behave like the generalization gap: they increase with the training set size (it is in the denominator but also hidden in the numerator, e.g., with the norm of the weights), while the generalization gap decreases.



Deep learning models can learn complex boundaries, which hurt uniform convergence bounds [and look like a bonanza for adversarial attackers] but not generalization.

Estimation of the Lipschitz constant of deep neural networks (M. Fazlyab et al.) (NeurIPS 2019)

Computing the Lipschitz constant of a neural net is NP hard: we can only hope to find approximations or bounds. Taking the product of the (matrix) norms of the weights is overly conservative. The Lipschitz constant can be expressed as the solution of a nonconvex optimization problem whose relaxation is a semidefinite program (solved with CVX and Mosek).

Robustness is one application: low-Lipschitz networks are more robust – conversely, networks trained with robust algorithms have a lower Lipschitz constant.

Data-dependent sample complexity of deep neural networks via Lipschitz augmentation C. Wei et al. (NeurIPS 2019)

Regularizers can be designed in a principled way, based on theoretical upper bounds of the generalization error, but those bounds are loose. The following data-dependent bound

$$\text{generalization} \leq \frac{\text{jacobian norm} \times \text{hidden layer norm}}{\text{margin} \times \sqrt{\text{training set size}}} + \text{lower-order terms}$$

where the Jacobian norm is the L^∞ norm of the Jacobian wrt the weights, the hidden layer norm is the L^∞ norm of the activations (on the training set), the margin is minimum, over the training set, of the difference between the first two logits, suggests to use the *Jacobian* as a regularizer (for the hidden layer norm, just use BatchNorm or LayerNorm).

Scalable global optimization via local Bayesian optimization D. Erikson et al. (NeurIPS 2019)

Thompson sampling can be used as an acquisition function for Bayesian optimization: sample a function from the posterior and estimate its minimum [?]. It is parallelizable: just sample k functions. In higher dimensions (beyond 10), do not try to find a global approximation of the objective, but just a *local* one.

Expand/shrink the region based on progress (trust-region-based Bayesian optimization, TuRBO).

Code available.

Uncertainty on asynchronous time event prediction M. Biloš et al. (NeurIPS 2019)

The forecast for the next event should change with time and account for uncertainty (it should not be a point in the simplex, but a distribution on it). Model continuously evolving distributions over the simplex with:

- Dirichlet distribution parameters evolving with a basis function decomposition;
- Logistic-normal parameters evolving with a weighted Gaussian process.



Applications include anomaly detection.

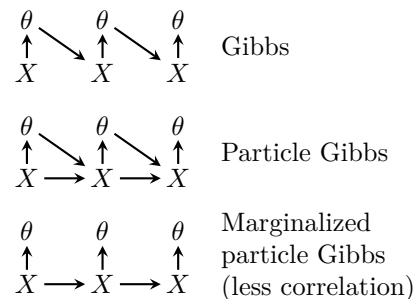
Variational Bayesian optimal experimental design A. Foster et al. (NeurIPS 2019)

An informative experimental design reduces entropy. The expected information gain $\text{EIG}(d) = E_y[H[P(\theta)] - P[p(\theta|y, d)]]$ is intractable: use variational inference.

Poisson mini-batching for Gibbs sampling with convergence rate guarantees R. Zhang and C. DeSa (NeurIPS 2019)

When using Gibbs sampling on subsets of variables, *i.e.*, when sampling from $x_i|x_J$ instead of $x_i|x_I$, $J \subsetneq I$, a Metropolis-Hastings rejection step is needed to correct the bias. Instead, add an *auxiliary Poisson variable* for each factor to control if it is used – it is unbiased, even without MH.

Parameter estimation in particle Gibbs sampling A. Wigren et al. (NeurIPS 2019)



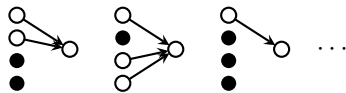
To fit SEIR (suceptible-exposed-infected-recovered) models for humans and mosquitoes, use Gibbs sampling: sample from $x|\theta, y$, then $\theta|x, y$. Sampling

from $x|\theta, y$ is difficult: use a particle filter. To reduce the correlation, marginalize what can easily be marginalized (conjugate distributions), either by hand or using a *probabilistic programming* such as `birch`, which uses a particle filter with delayed sampling.

Causal confusion in imitation learning
P. de Haan et al. (NeurIPS 2019)

A self-driving car, trained with imitation learning, could learn to pay attention to the brake indicator to decide when to brake, rather than the road... Adding more information (dashboard, history, etc.) can increase causal confusion and lead to poorer real-world performance.

Compute a latent state with a VAE; consider all possible causal graphs from the latent stated, encoded by binary masks; train each of them (train them together, with the mast as input). Run the models on new data (via interventions: change some past action and see what happens) and pick the best (look at the states independently).



Learning to control self-assembling morphologies: a study of generalization via modularities
D. Pathak et al. (NeurIPS 2019)

Dynamic graph networks combines several identical neural nets (with shared weights) into a graph, the output of one being sent to the input of the next ones.

Graded meta policy search
R. Mendonca et al. (NeurIPS 2019)

In meta-learning, learn *local policies*, for the training tasks, and combine them with supervised learning to deal with the whole tasks.

Using a logarithmic mapping to enable lower discount factors in reinforcement learning
H. van Seijen et al. (NeurIPS 2019)

In practice (with function approximation) only discount factors close to 1 work well. Poor performance seems to come from large differences in the action gaps between states

$$\Delta(s) = Q^*(s, a^*) - Q^*(s, \tilde{a}),$$

where a^* is the optimal action and \tilde{a} that chosen by the algorithm.

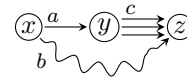
Logarithmic Q-learning replaces Q with $\log Q$; it is possible to get the correct optimum in spite of $E[\log X] \leq \log EX$.

Hindsight credit assignment

A. Harutyunyan et al. (NeurIPS 2019)

Do not use time for credit assignment, estimate $P(a|x, f(\tau))$, where a is a past action, x a past state and $f(\tau)$ a future outcome.

$$Q(x, a) = r(x, a) + \mathbb{E}_{\tau} \left[\sum \gamma^k \frac{P(a|x, X_k)}{\pi(a|x)} R_k \right]$$



$$\begin{aligned} P(a|x, y) &> P(b|x, y) && \text{more direct path} \\ P(c|y, z) &= \pi(c|y) && \text{any action will do} \end{aligned}$$

Weight-agnostic neural networks
A. Gaier and D. Ha (NeurIPS 2019)

Look for neural network architectures that work well without training.

To simplify the search, use a shared weight (yes, only one parameter: all the weights of all the layers are the same), random. Start with minimal networks of the form



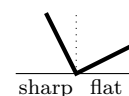
and modify them by inserting nodes, adding connections, or changing activation functions.

The performance is decent on reinforcement learning problems, competitive with SOTA models after training, for much smaller networks.

Those architectures contain implicit priors.

Asymmetric valleys: beyond sharp and flat local minima
H. He et al. (NeurIPS 2019)

According to popular belief, flat minima generalize better. But the loss landscape could be flat in some directions and sharp in others.



Prefer solutions biased towards the flat side – they generalize better; they can be obtained by averaging the solutions on the SGD path.

**UniXGrad: a universal, adaptive algorithm
with optimal guarantees
for constrained optimization**
A. Kavis et al. (NeurIPS 2019)

Gradient descent

$$x_{n+1} = x_n - \alpha \nabla f(x_n)$$

could be replaced by an “implicit” step,

$$x_{n+1} = x_n - \alpha \nabla f(x_{n+1})$$

which can be approximated with two explicit steps

$$\begin{aligned} y_{n+1} &= x_n - \alpha \nabla f(x_n) \\ x_{n+1} &= x_n - \alpha \nabla f(x_{n+1}). \end{aligned}$$

The *MirrorProx* algorithm adds a generalized projection wrt a Bregman divergence.

$$\begin{aligned} y_t &= \underset{y}{\operatorname{Argmin}} \langle y, \nabla f x_{t-1} \rangle + \alpha D_R(y, x_{t-1}) \\ x_t &= \underset{x}{\operatorname{Argmin}} \langle x, \nabla f y_t \rangle + \alpha D_R(x, x_{t-1}) \\ D_R(x, y) &= Rx - Ry - \langle \nabla R y, x - y \rangle \\ R(x) &= \frac{1}{2} \|x\|^2 \end{aligned}$$

The UniXGrad algorithm evaluates the gradients $\nabla f x_{t-1}$ and $\nabla f y_t$ at weighted averages \bar{x}_{t-1} and \bar{y}_t of the previous points $y_1, \dots, y_{t-1}, x_{t-1}$ and y_1, \dots, y_t . and uses an adaptive learning rate, defined with the sum of the gradient differences $\|\nabla f \bar{x}_{t-1} - \nabla f \bar{y}_t\|^2$.

**Sinkhorn barycenters with free support
via Frank-Wolfe algorithm**
G. Luise et al. (NeurIPS 2019)

To compute the average between probability distributions (for the Sinkhorn divergence, *i.e.*, entropic optimal transport), assume $a^* = \sum a_i \delta_{x_i}$, with the x_i ’s fixed and estimate the a_i ’s. Instad, do not fix the x_i ’s, but add them one by one, and do not re-estimate the previous coefficients.

Learning dynamic polynomial proofs
A. Fawzi et al. (NeurIPS 2019)

Given a polynomial P , prove that $\forall x \in [0, 1]^n P(x) \geq 0$, using a reinforcement learning agent, trained with DQN, with unsupervised dense rewards, accounting for symmetries.

**Generative modeling by estimating
gradients of the data distribution**
Y. Song and S. Ermon (NeurIPS 2019)

Generative models are usually implicit, only providing a sampling procedure (e.g., GANs) or explicit, directly modeling a probability distribution function (e.g., VAE), but these need to be normalized. Instead,

model the *score function* $\nabla_x \log p(x)$. It can be estimated from data (integration by parts, score matching).

$$\begin{aligned} \text{Fisher divergence} &= \frac{1}{2} \mathbb{E}_{x \sim \text{data}} \|\nabla_x \log p_{\text{data}}(x) - s_\theta(x)\| \\ &= \mathbb{E}_{x \sim \text{data}} \left[\frac{1}{2} \|s_\theta(x)\|_2^2 + \operatorname{tr} \nabla_x s_\theta(x) \right] \end{aligned}$$

It is necessary to add some noise:

- At least a very small amount, because the data lies on a low-dimensional manifold (or, even, is discrete), to help convergence;
- A larger amount of noise to have a better estimate of the score in low-density regions – but this leads to noisy samples.

The *noise-conditional score network* trains a single model for all levels of noise (the level of noise is one of the inputs). One can sample from a score function using *Langevin dynamics*; progressively reduce the amount of noise while sampling (*annealed Langevin dynamics*).

**Implicit generation and modeling
with energy-based models**
Y. Du and I. Mordatch (NeurIPS 2019)

Energy-based models (EBM)

$$p_\theta(x) \propto \exp -E_\theta(x)$$

can be trained with contrastive divergence

$$\begin{aligned} \mathcal{L}(\theta) &= \mathbb{E}_{x \sim \text{data}} [-\log p_\theta(x)] \\ \nabla \mathcal{L} &= \mathbb{E}_{x \sim \text{data}} [\nabla E_\theta(x)] - \mathbb{E}_{x \sim p_\theta} [\nabla E_\theta(x)] \end{aligned}$$

and sampled from with Langevin dynamics.

**Residual flows
for invertible generative modeling**
R.T.Q. Chen et al. (NeurIPS 2019)

The *invertible ResNet* $f : x \mapsto x + g(x)$ is invertible if g has Lipschitz constant less than 1 (enforced by spectral normalization) and its inverse can be computed by fixed point iterations. To compute

$$\log |\det \nabla_x f| = \sum \frac{(-1)^k}{k} \operatorname{tr} J_g(x)^k,$$

use the **Russian roulette estimator** which replaces the infinite sum with an unbiased estimator, computed as a weighted sum of a random number of terms.

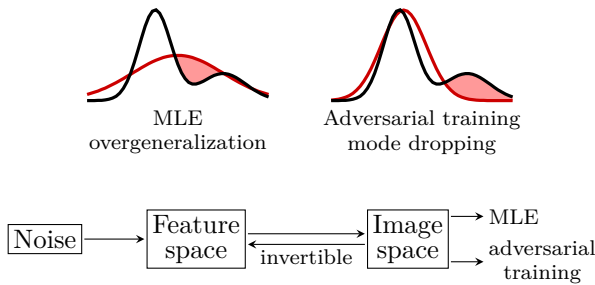
The *lipswish* activation is Lipschitz.

$$\begin{aligned} \text{swish}(x) &= x \cdot \text{sigmoid}(x) \\ \text{lipswish}(x) &= \text{swish}(x)/1.1 \end{aligned}$$

Hype: a benchmark for human eye perceptual evaluation of generative models
S. Zhou et al. (NeurIPS 2019)

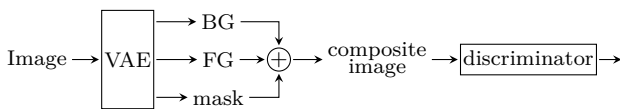
Use crowdsourcing to evaluate the quality of a generative model, by measuring the time needed by humans to decide if an image is real or fake, either by showing the image for a limited amount of time, progressively increased or decreased depending on performance (“adaptive staircase procedure”, from psychometrics) or by letting the user choose how long they need. It costs \$60 and takes 60 minutes per model. It was tested on WGAN-GP, ProGAN, SN-GAN, BEGAN, BigGAN.

Adaptive density estimation for generative models
T. Lucas et al. (NeurIPS 2019)



Maximum likelihood estimators (MLE) and adversarial training have complementary advantages and drawbacks: combine them into a hybrid approach to ensure both dataset coverage and sample quality.

Emergence of object segmentation in perturbed generative models
A. Bielski and P. Favaro (NeurIPS 2019)



Unsupervised object segmentation is possible: add a random shift to the foreground and mask: the reconstructed image should still look real.

Faster width-dependent algorithm for mixed packing and covering LPs
D. Boob et al. (NeurIPS 2019)

A mixed packing-covering linear program (MPC-LP) is of the form

$$\begin{array}{ll} \text{Find} & x \in [0, 1]^n \\ \text{Such that} & Px \leq \mathbf{1} \\ & Cx \geq \mathbf{1} \quad \text{where } P, C \geq 0. \end{array}$$

The corresponding *saddle point problem* is

$$\text{Minimize}_{x \in [0, 1]^n} \text{Max}_{\substack{y \in \Delta_p \\ z \in \Delta_q}} L(x, y, z).$$

It is often solved by minimizing the *primal-dual gap* f , with a strong convex penalty ϕ ,

$$\text{Minimize}_w f(w) + \phi(w)$$

with *Nesterov's dual extrapolation*. Replace strong convexity

$$\phi\left(\frac{u+v}{2}\right) \leq \frac{1}{2}[\phi(u) + \phi(v)] - \frac{1}{2}\|u-v\|$$

with a weaker notion, area convexity,

$$\phi\left(\frac{u+v+t}{3}\right) \leq \frac{1}{3}[\phi(u) + \phi(v) + \phi(t)] - \frac{1}{3\sqrt{3}}(v-u)'M(u-t).$$

For instance, $\gamma(x, y) = xy \log x + 2y \log y$, on $[0, 1]^2$.

Sparse logistic regression learns all discrete pairwise graphical models
S. Wu et al. (NeurIPS 2019)

Logistic regression, with an ℓ_1 (or $\ell_{2,1}$) penalty, can recover the structure of a graphical model, such as the Ising (binary) model

$$P(z) \propto \exp \left[\sum_{i,j} A_{ij} z_i z_j + \sum_i \theta_i z_i \right] \quad z \in \{\pm 1\}^n.$$

Smoothing structured decomposable circuits
A. Shih et al. (NeurIPS 2019)

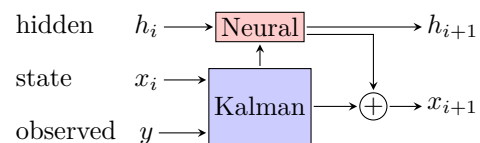
Circuits (aka *sum-product networks*) describe probability distributions and allow efficient computation of likelihood partition function, marginals, etc. provided they are smooth.

A circuit is *smooth* if the children of any \oplus gate use the same variables. It is easy to ensure, but the algorithm is quadratic; it can be sped up by representing groups of variables as intervals.

Tractable probabilistic models
V. van den Broeck et al. (UAI 2019)

Combining generative and discriminative models for hybrid inference
V.G. Satorras et al. (NeurIPS 2019)

Use a neural network to add a correction term to Kalman-like algorithms.



Fast and accurate least mean squares
A. Maalouf et al. (NeurIPS 2019)

The average of n points in \mathbf{R}^d is a convex combination of $d + 1$ of those points; such a representation can be obtained recursively, by putting the points in $d + 1$ (random) groups and computing their means (Caratheodory theorem).

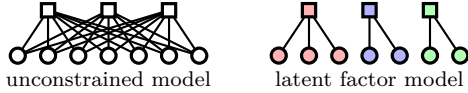
Transform the least mean square (LMS) problem $\text{Argmin}_x \|Ax - b\|_2^2 + g(x)$:

$$\|Ax - b\|_2^2 = [x - 1] \begin{bmatrix} A' \\ b' \end{bmatrix} [A \ b] \begin{bmatrix} x \\ -1 \end{bmatrix} = [x - 1] P' P \begin{bmatrix} x \\ -1 \end{bmatrix}.$$

The covariance matrix $\frac{1}{n} P' P = \frac{1}{n} \sum p_i p_i'$ is the average of n points, $p_i p_i'$ in \mathbf{R}^{d^2} : it can be replaced with a convex combination of $d^2 + 1$ points (useful if $d^2 + 1 \ll n$)

Fast structure learning with modular regularization
G. VerSteeg et al. (NeurIPS 2019)

To recover the dependence structure among a large number of variables from a small number of observations, assume that each observed variable has a single latent parent.



The *total correlation*

$$\begin{aligned} \text{TC}(\mathbf{X}) &= \sum_i H(X_i) - H(\mathbf{X}) \\ &= \text{KL}(\text{joint} \parallel \text{product of marginals}) \end{aligned}$$

aka multivariate mutual information (it generalizes $H(X; Y) = H(X) + H(Y) - H(X, Y)$) can be used to characterize graphical models such that

$$\begin{aligned} \forall i \neq j \quad Z_i \perp\!\!\!\perp Z_j & \quad (\text{latent}) \\ \forall i \neq j \quad X_i \perp\!\!\!\perp X_j \mid \mathbf{Z} & \quad (\text{observed}) \end{aligned}$$

by $\text{TC}(X|Z) + \text{TC}(Z) = 0$. For a “modular” model (each X_i depends on a single Z_j), it suffices to add $\forall i \text{TC}(Z|X_i) = 0$.

To fit those models, set $Z = WX + \varepsilon$, $\varepsilon \sim N(0, \text{diag}(\eta))$ and solve

$$\underset{W}{\text{Minimize}} \text{TC}(X|Z) + \text{TC}(Z) + \text{penalty},$$

where the penalty makes the $\text{TC}(Z|X_i)$ small and cancels out computationally expensive terms.

This can also be applied to large correlation matrices in finance: we recover industry classifications.

Numpy and Pytorch code available: **LinearCorex**, **T-Corex**.

Principal component projection and regression in nearly linear time through asymmetric SVRG
Y. Jin and A. Sidford (NeurIPS 2019)

Principal component projection and PCR are usually done by explicit computation of the eigenvectors or through polynomial approximations of the sign function. Instead, use a *rational approximation* (Zolotarev). It is also possible to reduce the variance of the estimates by expanding the eigenvalue problem to higher dimensions. The algorithm is almost linear in $\text{nnz}(A)$.

PIDForest: anomaly detection via partial identification
P. Gopalan et al. (NeurIPS 2019)

The *sparsity* of a region C is

$$\text{sparsity}(C) = \frac{\text{volume}(C)}{\# \text{ points in } C}.$$

Anomalies can be partially identified (PID) with their PID score:

$$\text{score}(x) = \max_{\substack{C \in \mathcal{C} \\ x \in C}} \text{sparsity}(C),$$

which can be efficiently approximated with a random forest.

A neurally plausible model learns successor representation in partially observable environments
E. Vertez and M. Sahani (NeurIPS 2019)

The **successor representation** (SR), the expected discounted future state occupancy,

$$M(s, s') = E \left[\sum_{k \geq 0} \gamma^k \mathbf{1}_{s_{t+k}=s'} \mid s_t = s \right]$$

is a hybrid between model-based and model-free RL. The *successor features* replace state occupancy with state features.. If the reward function changes, one can use the SR to compute the value function – but it does not work with noisy observations. To represent the distribution (belief) on the current state, use the *distributed distributional code* (DDC)

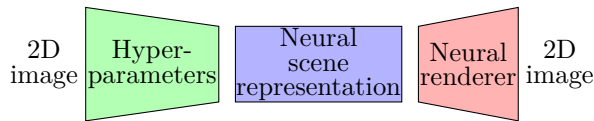
$$\mu = E_{s \sim p} [\psi(s)] \quad M(\mu) = E \left[\sum_{k \geq 1} \gamma^k \psi * s_{t+k} \mid \mu_t = \mu \right]$$

Learning reward machines for partially observable RL
R.T. Icarte et al. (NeurIPS 2019)

A *reward machine* (RM) is an automaton-based reward function. It can be used as a memory for POMDP problems. Learning a RM is a discrete optimization problem.

Scene representation networks: continuous 3D structure aware neural scene representation
V. Sitzmann et al. (NeurIPS 2019)

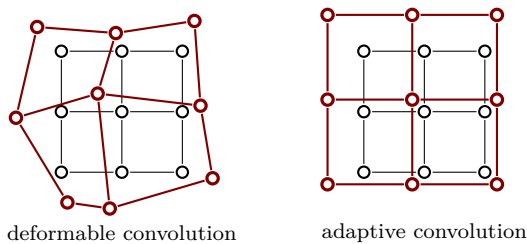
From a 2D image, use a *hypernetwork* to learn a *scene representation network*, which can be fed to a *neural renderer* to generate images from other points of view. The SRN maps positions to the properties (colour, shape, etc.) at that position/voxel; contrary to voxel-based representations there are no discretization artefacts. The neural renderer uses *ray marching* (look for objects closest to the camera on each ray from the camera), sped up by predicting the distance to the nearest object from the camera in each direction.



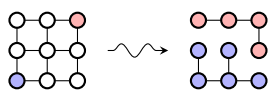
A condition number for joint optimization of cycle-consistent networks
L.J. Guibas et al. (NeurIPS 2019)

A *mapping graph* is a graph (V, E) , with a set D_u associated to each vertex and a mapping $D_u \rightarrow D_v$ associated to each edge $u \rightarrow v$. It is *cycle-consistent* if the composition of the mappings along any cycle is the identity (as with CycleGAN or back-translation).

Cascade RPN: delving into high quality region proposal network with adaptive convolution
T. Vu et al. (NeurIPS 2019)



Probabilistic watershed: sampling all spanning forests for seeded segmentation and semi-supervised learning
E.F. Sanmartín et al. (NeurIPS 2019)



Poincaré recurrence, cycles and spurious equilibria in gradient descent-ascent for non-convex non-concave zero-sum games
E. Vlatakis (NeurIPS 2019)

Gradient descent-ascent does not always work or non-convex non-concave zero-sum games (GANs, etc.): they suffer from spurious equilibria, cycles, and divergence.

Graph-based discriminators: sample complexity and expressiveness
R. Livni and Y. Mansour (NeurIPS 2019)

To compare two distributions from their samples, use and *integral probability measure* (IPM)

$$\text{IPM}(p_1, p_2) = \sup_{h \in H} \left| \mathbb{E}_{x \sim p_1} h(x) - \mathbb{E}_{x \sim p_2} h(x) \right|, \quad H \subset \{0, 1\}^X$$

or a more powerful *graph-based discriminator*

$$\text{IPM}(p_1, p_2) = \sup_{g \in G} \left| \mathbb{E}_{x, y \sim p_1} g(x, y) - \mathbb{E}_{x, y \sim p_2} g(x, y) \right|,$$

for $G \subset \{0, 1\}^{X \times X}$.

Learning by abstraction: the neural state machine
D. Hudson and C. Manning (NeurIPS 2019)

Neural nets can get the right answer for the wrong reason. The *neural state machine* is a differentiable graph-based model outputting a *scene graph*, used as a state machine, with objects (with attribute – colour, shape, etc – given by several embeddings) as nodes and relations (e.g., relative positions) as edges. A question about the scene (visual question answering, VQA) is translated into a sequence of instructions, which are then run on the state machine.

Batched multi-arm bandits
Z. Gao (NeurIPS 2019)

In the batched multi-arm bandit problem, one chooses k arms, gets the results, and plays again. The grid (k) can be imposed or chosen.

Are sample means in multi-arm bandits positively or negatively biased?
J. Shin et al. (NeurIPS 2019)

The sample mean (of a fixed arm, at a fixed time) is negatively biased for ϵ -greedy, UCB, Thompson thresholding. But for the chosen arm and a stopping time:

- Negatively biased under optimistic sampling;
- Positively biased under optimistic stopping;
- Positively biased under optimistic choosing.

SIC-MMAB: synchronization involves communication in multi-player multi-arm bandits
E. Boursier (NeurIPS 2019)

In the *multi-player MAB*, the reward is zero in case of a collision. If the collisions are observed and the players can communicate, the regret for M players is

$$\sum_{k > M} \frac{\log T}{\mu_M - \mu_k}.$$

If they cannot communicate, this does not change: if they are synchronized, they can use the collisions to exchange bits of information. If they are not synchronized (if they do not start at the same time) and do not observe the collisions (only the zero rewards), this is trickier.

Recovering bandits

C. Pike-Burke and S. Grunewalder (2019)

Bandits are often used for product recommendation, but you do not want to recommend a sofa to someone who has just bought one – you should wait. In *recovering bandits*, the reward is a function of the time since the arm was last played, modeled as a Gaussian process. They require some strategic thinking: if you expect a good arm's reward to increase, you may want to wait and play a less good arm for the time being. UCB and Thompson sampling can be modified into d -step look-ahead strategies.

Strategizing against no-regret learners

Y. Deng et al. (NeurIPS 2019)

Mean-based algorithms can be fooled: play a suboptimal action long-enough to trick your opponent into thinking it is optimal for you, then play the actual optimal solution for as long as it takes your opponent to realize they were duped.

No-regret algorithms include EXP3, multiplicative weights, follow the perturbed leader.

Time-accuracy tradeoffs for learning a ReLU with respect to Gaussian marginals

S. Goel et al. (NeurIPS 2019)

ReLU regression looks for w such that $y \approx \text{relu}(w'x)$. If $x \sim N(0, 1)$, a shallow neural net with stochastic gradient descent learns w .

Small ReLU networks are memorizers: a tight analysis of memorization capacity

C. Yun et al. (NeurIPS 2019)

A 2-layer fully-connected ReLU network can memorize N points with $\Theta(\sqrt{N})$ nodes (vs $\Theta(N)$ for a 1-layer neural net).

On making stochastic classifiers deterministic

A. Cotter et al. (NeurIPS 2019)

To solve a non-convex constrained problem, e.g. with fairness constraints

$$\begin{array}{ll} \text{Find} & \theta \\ \text{To minimize} & g_0(\theta) \\ \text{Such that} & \forall i g_i(\theta) \leq 0 \end{array}$$

look for Nash equilibrium of the Lagrangian

$$\mathcal{L}(\theta, \lambda) = g_0(\theta) + \sum \lambda_i g_i(\theta).$$

There might be no pure Nash equilibrium, but there is a mixed one, giving rise to a *stochastic classifier* (alternatively update θ and λ , and put a uniform distribution on the trajectory).

To transform a stochastic classifier into a deterministic one, one could use thresholding $\hat{f} = \mathbf{1}_{f \geq 1/2}$, but the performance can be very different (for instance if $f \equiv 0.51$). Instead, use *hashing* (cf the difference between thresholding and dithering for images), which fakes stochasticity – it replaces stochasticity with arbitrariness.

One can try to interpolate between thresholding and hashing: partition the samples into different regions and use a different threshold in each.

Cold case: the lost MNIST digits

C. Yadov and L. Bottou (NeurIPS 2019)

MNIST was not the whole data: 50,000 samples were discarded (computers were less powerful, then): aMNIST adds them back and provides some metadata.

Testing set rot is real: MNIST classifiers perform less well on the new data than on the initial dataset – however, the ranking of the classifiers remains unchanged.

Optimal sparse decision trees

X. Hu et al. (NeurIPS 2019)

It is possible to compute the *optimal sparse decision tree* (with a penalty on the number of nodes), in time (often) linear in the number of samples and features, with a proof of optimality (in time exponential in the number of features). Code available.

Optimizing generalized rate metrics with three players

H. Narasimhan et al. (NeurIPS 2019)

We often need to train models with *fairness constraints*, e.g., $|\text{precision}(G_1) - \text{precision}(G_2)| \leq \varepsilon$,

$$\begin{array}{ll} \text{Find} & \theta \\ \text{To minimize} & \psi(R_1(\theta), \dots, R_k(\theta)) \\ \text{Such that} & \forall i \psi_i(R_1(\theta), \dots, R_k(\theta)) \leq 0 \end{array}$$

where the R_k 's are prediction rates, *i.e.*, expectations of counts (FPR, etc.) on subsets of the data. Introduce slack variables

$$\begin{array}{ll} \text{Find} & \theta, \xi \\ \text{To minimize} & \psi(\xi_1, \dots, \xi_k) \\ \text{Such that} & \forall i \xi_i \geq R_i(\theta). \end{array}$$

The Lagrangian formulation

$$\begin{array}{ll} \text{Minimize} & \psi(\xi) - \lambda'(\xi - R(\theta)) \\ \theta, \xi & \lambda \geq 0 \end{array}$$

is a 3-player game: use the exact solution for x_i , SGD for λ , and SGD with surrogates for $R_k(\theta) \leq \tilde{R}_k(\theta)$.

There is a TensorFlow constrained optimization (TFCO) library.

***This looks like that: deep learning
for interpretable image recognition***
C. Chen et al. (NeurIPS 2019)

Attention is not sufficient to explain what a convnet sees: it only tells where, not what. Add a *prototype layer*, which learns prototypes and outputs how similar each prototype is to a given patch; combine the prototype scores into output logits.

Paradoxes in fair machine learning
(NeurIPS 2019)

Fairness in machine learning is often defined as some quantity, e.g., odds, being equal among all groups (for some predefined protected attributes). Fairness in *fair division* relies on a few axioms, e.g.,

- Resource monotonicity: adding more resources makes everyone better off;
- Population monotonicity: adding more people makes everyone worse off.

Adding resource monotonicity to equal odds does not impact efficiency; adding population monotonicity does.

***Multicriteria dimensionality reduction
with applications to fairness***
T. Tantipongpipat et al. (NeurIPS 2019)

PCA is not fair: the reconstruction error need not be the same for all groups (for instance, face reconstruction for males is easier than for females). Instead, consider optimization problems of the form

$$\text{Minimize}_{P \text{ projection}} g(f_1(P), \dots, f_k(P))$$

where $f_i(P)$ is the reconstruction error for group i and g some aggregation function (e.g., sum, for PCA). For instance

$$\begin{aligned} &\text{Minimize}_P \text{Max}_{i \in [1, k]} \left(\text{Max}_Q \|A_i Q\|_F^2 - \|A_i P\|_F^2 \right) \\ &\text{Maximize}_P \prod \|A_i P\|_F^2. \end{aligned}$$

***Implicit regularization
in deep matrix factorization***
S. Arora et al. (NeurIPS 2019)

Deep neural networks generalize well, even with no regularization – it is widely believed that SGD provides implicit regularization.

The matrix completion problem is often solved with a nuclear norm penalty

$$\text{Minimize}_W \sum_{(i,j) \in \Omega} (w_{ij} - b_{ij})^2 + \lambda \|W\|_*$$

but a deep matrix approximation (yes, a deep *linear* network) performs better

$$\text{Minimize}_{W_1, \dots, W_N} \sum_{(i,j) \in \Omega} [(W_N W_{N-1} \cdots W_1)_{ij} - b_{ij}]^2$$



(no regularization).

In deeper networks, small singular values move more slowly with gradient descent, while larger singular values move faster: the longer gaps between singular values give a low-rank matrix.

***SGD on neural networks learns
functions of increasing complexity***
D. Kalimeris et al. (NeurIPS 2019)

Stochastic gradient descent (SGD) starts to learn an “essentially linear” classifier (its performance can be explained by a linear classifier) and then increasingly complex models.

When does label smoothing help?
R. Müller et al. (NeurIPS 2019)

Label smoothing replaces hard targets  with soft targets : this ensures that observations are close to their prototypes (weights of the last layer) and far from the others – the clusters are tighter.

Neural networks are over-confident: temperature scaling and label smoothing can help calibrate them.

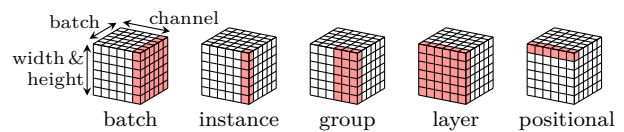
Label smoothing is bad for distillation: the model no longer knows how different the classes are.

***Splitting steepest descent
for growing neural architectures***
L. Wu et al. (NeurIPS 2019)

To build smaller neural nets, progressively split neurons into several copies.

Positional normalization
B. Li et al. (NeurIPS 2019)

Normalization variants:



The moments (μ, σ) used for the normalization contain information: re-inject them later in the network.

Variance reduction for matrix games
Y. Carmon et al. (NeurIPS 2019)

In zero-sum games

$$\text{Min}_{x \in \mathcal{X}} \text{Max}_{y \in \mathcal{Y}} f(x, y),$$

\mathcal{Y} can be seen as constraints (GAN) or uncertainty (robust optimization).

Examples of bilinear games $\text{Min}_x \text{Max}_y y' A x$ include *matrix games* ($\mathcal{X} = \mathcal{Y} = \text{simplex}$), SVM (\mathcal{X} Euclidean, \mathcal{Y} simplex) and linear regression ($\mathcal{X} = \mathcal{Y} = \text{Euclidean}$).

**Provably robust deep learning
via adversarially trained smoothed classifiers**
H. Salman et al. (NeurIPS 2019)

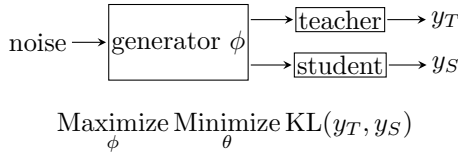
Randomized smoothing transforms a classifier f into

$$g(x) = \underset{y \in \mathcal{Y}}{\operatorname{Argmax}} \mathbb{P}_{\delta \sim N(0, \sigma^2 I)} [f(x + \delta) = y].$$

Look for adversarial examples for the smoothed classifier.

**Zero-shot knowledge transfer
via adversarial belief matching**
P. Micaelli and A. Storkey (NeurIPS 2019)

Model compression (pruning, quantization, distillation) usually requires training data, but one can use a generator instead. This works even though the generated samples do not look like actual samples – they suffice to describe the teacher’s decision boundaries.



Learning perceptual inference by contrasting
C. Zhang et al. (NeurIPS 2019)

Raven’s progressive matrices (RPM) are psychological tests for spatial-temporal reasoning (“thinking in pictures”, for humans). With computer-generated datasets (RAVEN, PGM), computers can have a go at it, modeling the attributes in the images and the rules they follow with a *Gumbel softmax*.

**Better transfer learning
with inferred successor maps**
T. Madarasz and T. Behrens (NeurIPS 2019)

Replace the Q -value function

$$Q_t^\pi(s, a) = \mathbb{E}[\sum \gamma^k r_{t+k} | s_t = s, a_t = a]$$

with the *successor representation* (SR)

$$M_t^\pi = \mathbb{E}[\sum \gamma^k \mathbf{1}_{s_{t+k+1}=s'} | s_t = s, a_t = a]$$

$$Q^\pi(s,) = \sum_{s'} M(s, a, s') w(s')$$

**Infra-slow brain dynamics
as a marker of cognitive function and decline**
S. Ajmera (NeurIPS 2019)

Gaussian process factor analysis (GPFA) can extract latent signals, at various time scales, from fMRI data.

y : high-dimensional signal (observed)
 x : low-dimensional signal (latent)
 $y = Ax + \text{noise}$
 $x_i \sim \text{GP}$

Understanding sparse JL for feature hashing
M. Jagadeesan (NeurIPS 2019)

Sparse Johnson-Lindenstrauss is a dimension reduction method trying to preserve distances, defined by a random transformation, with s nonzero entries in each column. Equivalently, it is the sum of s feature hashing dimension reductions.

$$h : \llbracket 1, n \rrbracket \rightarrow \llbracket 1, m \rrbracket \text{ hash function}$$

$$\sigma_i \in \{\pm 1\} \text{ random}$$

$$f(x) = \sum_{j \in h^{-1}(i)} \sigma_j x_j$$

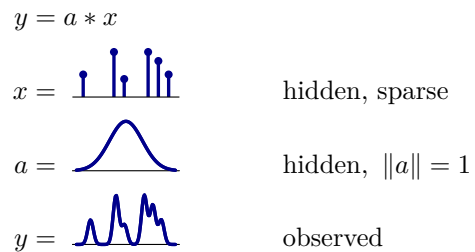
There is a tradeoff between the number of hash functions s , the dimension m , and ℓ^2 distance preservation with 3 regimes:

- m small: poor performance, regardless of the number of hash functions;
- m large: performance proportional to \sqrt{s} ($s = 4$ is a good choice);
- m very large: good performance, regardless of the number of hash functions.

The performance is measured by looking at vectors with small ℓ_∞ -to- ℓ_1 norm ratio.

**A non-convex approach for exact and efficient
multichannel sparse blind deconvolution**
Q. Qu et al. (NeurIPS 2019)

Gradient descent, Huber loss and preconditioning efficiently solve the blind deconvolution problem.



Applications include superresolution (microscopy) and neuron spike identification.

**Efficiently learning
Fourier sparse set functions**
A. Amrollahi et al. (NeurIPS 2019)

The Walsh-Hadamard (or Fourier) transform of a *set function* $x : 2^V \rightarrow \mathbf{R}$, seen as a function $x : \mathbf{F}_2^n \rightarrow \mathbf{R}$, is

$$\hat{x} : \begin{cases} \mathbf{F}_2^n & \rightarrow \mathbf{R} \\ f & \mapsto \sum_{t \in \mathbf{F}_2^n} (-1)^{\langle f, t \rangle} x_t. \end{cases}$$

It is *sparse* if $\|\hat{x}\|_0 \leq k$, and low-frequency if $\forall f \in \operatorname{supp} \hat{x} \ |f| \leq d$, where $|f|$ is the Hamming weight of f , *i.e.*, the number of nonzero elements.

For instance, the *cut function* of a graph,

$$x(A) = \sum_{\substack{s \in A \\ t \in V \setminus A}} w(s, t)$$

is sparse ($k = |E|$) and low-frequency ($d = 2$).

Applications include hyperparameter tuning, where $x \in 2^V$ is a boolean vector describing the hyperparameters (SGD vs Adam, number of layers, layer size, etc.) and $F(x)$ the corresponding loss.

Surfing: iterative optimization over incrementally trained deep networks
G. Song et al. (NeurIPS 2019)

To invert a generator (e.g. a VAE), do not start with the fully trained model, but follow a local optimum through the training (“surfing”) – the loss landscape is much nicer at the beginning.

R2D2: repeatable and reliable detector and descriptor
J. Revaud (NeurIPS 2019)

To align two images, identify the “keypoints”, compute their “descriptors” (e.g., SIFT), and match the points if they are close. We want the keypoints to be, not only repeatable (still present on a slightly modified image), but also reliable (*i.e.*, useful for matching – this excludes repeated patterns),

Training image estimators without image ground truth
Z. Xia and A. Chakrabarti (NeurIPS 2019)

To learn how to restore image quality, e.g., for motion deblurring of face images, when (low-quality, high-quality) image pairs are not available, use pairs of low-quality images corresponding to the same high-quality image,

$$\begin{aligned} y_1 &= \theta_1 x + \varepsilon_1 \\ y_2 &= \theta_2 x + \varepsilon_2 \end{aligned}$$

with a *swap loss* $\|\hat{\theta}_1 \hat{x}_2 + \hat{\varepsilon}_1 - y_1\|$.

KernelGAN: blind SR kernel estimation using an internal GAN
S. Bell-Kligler (NeurIPS 2019)

Deep *linear* network (equivalent to a 1-layer network, but with more local minima) to infer the kernel to use for super-resolution (SR).

Differentiable ranking and sorting using optimal transport
M. Cuturi et al. (NeurIPS 2019)

The 1-dimensional optimal transport can be solved by sorting. Conversely, the entropic regularization of the OT problem provides a differentiable regularization of the sorting and ranking operations.

Nonparametric density estimation and convergence rates for GANs under Besov IPM losses
A. Uppal et al. (NeurIPS 2019)

Density estimation does not work well in high dimensions, but GANs solve a similar problem. GANs minimize an *integral probability metric* (IPM)

$$\begin{aligned} \hat{P} &= \operatorname{Argmin}_Q \sup_{f \in \mathcal{F}} \mathbb{E}_{X \sim Q} [f(X)] - \mathbb{E}_{X \sim \text{data}} [f(X)] \\ &= \operatorname{Argmin}_Q f_{\mathcal{F}}(Q, \text{data}) \\ d_{\mathcal{F}}(P, Q) &= \sup_{f \in \mathcal{F}} \left| \mathbb{E}_{X \sim P} [f(X)] - \mathbb{E}_{X \sim Q} [f(X)] \right| \end{aligned}$$

Besov spaces $B_p^s \approx \{f \in L^p : \|f^{(s)}\| \text{ bounded}\}$ are generalization of Sobolev spaces ($p = 2$); they are well-approximated with neural nets with ReLU activations.

Wasserstein Weisfeiler-Lehman graph kernels
M. Togninalli et al. (NeurIPS 2019)

Traditional graph kernels decompose the graphs into simple substructures, compute the similarities between those substructures and aggregate them in a simplistic way, by averaging. Use the Wasserstein distance instead (or, rather, $e^{-\lambda d_W}$).

Putting an end to end-to-end
S. Löwe et al. (NeurIPS 2019)

Greedy InfoMax (GIM) is a form of semi-supervised (or even unsupervised) learning which learns a neural network without using end-to-end back-propagation (the forward pass is still end-to-end, though, but can be done less often; learning can be asynchronous) by dividing it into “modules” (groups of layers), each with its own “self-supervised” loss, InfoNCE, the mutual information between nearby patches (for images or sound).

On the downstream performance of compressed word embeddings
A. May et al. (NeurIPS 2019)

The *eigenspace overlap score* (EOS) measures how much information is retained when compressing word embeddings $A_{\text{raw}} \mapsto A_{\text{compressed}}$, from the left singular vectors, $A = UDV'$,

$$\text{EOS} = \frac{1}{d} \|U'_{\text{compressed}} U\|_F^2.$$

Numerically accurate hyperbolic embeddings using tiling-based models
T. Yu and C. de Sa (NeurIPS 2019)

Euclidean models of hyperbolic space are numerically inaccurate far from the origin: instead of using Euclidean coordinates, consider a tiling of hyperbolic space and represent a point by a tile (an element of

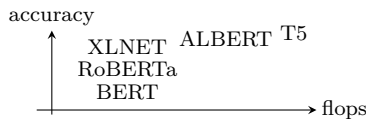
the tiling group) and the Euclidean coordinates of the corresponding point in the central tile (close to the origin and therefore accurate).

XLNet for language pretraining **Z. Yang et al. (NeurIPS 2019)**

Autoregressive language models are usually unidirectional: XLNet changes this order of the words, so that it becomes bidirectional

$$P[x_{i_{k+1}} | x_{i_1}, \dots, x_{i_k}, x_{i_{k+1}}].$$

The new position i_{k+1} can be used as an attention key.



A step towards quantifying independently reproducible ML research **E. Raff (NeurIPS 2019)**

A paper is *independently reproducible* if it is reproducible without the author's code. They tried to reproduce 255 papers, succeeded for 60% of them, and identified the features of reproducible papers.

Average individual fairness **A. Roth (NeurIPS 2019)**

Fairness is often defined from the false negative rate for groups of interest. Instead of looking at the negative rate for a single problem for each group, look at the negative rate over several problems for a single subject.

Optimal transport methodology for lineage tracing **K. Yang (NeurIPS 2019)**

To use optimal transport for *lineage tracing* (to study cancer progression or embryogenesis):

- Use (unbalanced, relaxed) *Monge* optimal transport, *i.e.*, learn a transport map (instead of a coupling), parametrized by a neural net: it is faster and provides a mapping for new samples;
- Use optimal transport wrt the Euclidean distance in a latent space, learned by an autoencoder.

Estimation of Wasserstein distances in the spiked transport model **J. Niled-Weed (NeurIPS 2019)**

To estimate the Wasserstein distance $W_p(\mu, \nu)$ from samples $X_1, \dots, X_n \sim \mu$, $Y_1, \dots, Y_n \sim \nu$, one is tempted to use the plugin estimator $W_p(\hat{\mu}, \hat{\nu}_n)$, but the difference is of the order $n^{1/d}$: the empirical distribution $\hat{\mu}_n$ is a bad estimator of μ . In some cases (e.g., when μ has a density), better estimates are available.

While it may be possible to estimate $W_p(\mu, \nu)$ directly rather than through estimates $\hat{\mu}$ and $\hat{\nu}$ of μ and ν , the difference is still at least $(n \log n)^{-1/d}$.

The *spiked covariance model* is $X \sim N(0, \Sigma)$, $\Sigma = I + \beta vv'$, where I is high-dimensional noise and $\beta vv'$ a low-dimensional perturbation.

The *spiked transport model* assumes

$$\begin{aligned} \text{supp } X, \text{supp } Y &\subset \mathcal{U} \text{ (low-dimensional)} \\ \text{supp } Z &\subset \mathcal{U}^\perp \\ \mu &= \text{Law}(X + Z) \\ \nu &= \text{Law}(Y + Z) \end{aligned}$$

and looks for a coupling between X and Y . The difference is

$$n^{-1/k} + \sqrt{\frac{d \log n}{n}}$$

(the second term corresponds to the search for \mathcal{U} .)

Generalizing learning with optimal transport **S. Jegelka et al. (NeurIPS 2019)**

One can align word embeddings for different languages with optimal transport after adding some rotational invariance:

$$\text{Minimize}_{\Gamma \in \Pi(\mu, \nu)} \min_{P \in \mathcal{F}} \sum_{ij} \Gamma_{ij} d(x_i, Py_j)$$

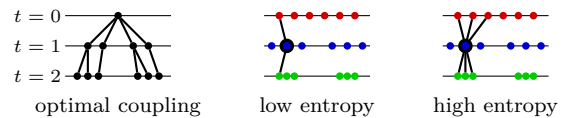
with $\mathcal{F} = \{P \in \mathbf{R}^{d \times d} : \|P\|_p \leq k\}$, where $\|\cdot\|_p$ is the *Shatten norm* (the L^p norm of the singular values).

Use entropic regularization, normalization, and stabilized Sinkhorn.

Regularize the adversary: it should be close to an orthonormal transformation (no extreme distortion), $\beta \min_{P \in O_n} \|f_W(X) - XP'\|^2$, computed with *orthogonal Procrustes*, $P^* = UV'$, where $f_W(X')X = U\Sigma V'$.

Towards a mathematical theory of development **G. Schiebinger (NeurIPS 2019)**

Single-cell RNA sequencing (scRNA-seq) provides snapshots if the genes expressed at various stages of development. Optimal transport can link those snapshots: *unbalanced* optimal transport can account for cell proliferation and entropic regularization.



$$\begin{aligned} \text{Find} \quad & \pi \\ \text{To minimize} \quad & \iint c(x, y) \pi(x, y) dx dy - \varepsilon H(\pi) \\ \text{Such that} \quad & \pi(\cdot, y) = P_{t_2}(y) \\ & \pi(x, \cdot) = P_{t_1}(y) g(x)^{t_2 - t_1} \end{aligned}$$

Model the time dependence as $\dot{x} = f(x)$:

$$\underset{f}{\text{Minimize}} \|x_{t_2} - f(x_{t_1})\|^2 + \text{reg}(f).$$

***Unsupervised hierarchy matching
with optimal transport over hyperbolic space***
D. Alvarez-Melis et al. (NeurIPS 2019)

To match hierarchies using optimal transport, accounting for orthonormal transformations is not enough: the order of the branches can change. Use continuous non-linear mappings, parametrized by a neural net.

***Optimal transport mapping
input convex neural networks***
A.V. Makkuva (NeurIPS 2019)

The optimal transport can be defined as $T = \nabla f^*$,

$$f = \underset{f \in \text{CVX}}{\text{Argmin}} \mathbb{E}_{X \sim P}[f(X)] + \mathbb{E}_{Y \sim Q}[f^*(Y)]$$

and f can be parametrized as an *input convex neural network* (ICNN).

Wasserstein style transfer
Y. Mroueh (NeurIPS 2019)

To perform style transfer, from a style image I_s to a content image I_c , learn an encoder-decoder pair, $I \mapsto (F_j(I))_j \mapsto I$, where j is the position, in the image, of the features F , and a Monge map T between the empirical distributions $\sum \delta_{F_j(I_c)}$ and $\sum \delta_{F_j(I_s)}$.

That map is learned using a Gaussian approximation

$$W_2(\mu, \nu) \geq W_2(N(\mathbb{E} \mu, \text{Var} \mu), N(\mathbb{E} \nu, \text{Var} \nu)).$$

The Wasserstein distance and the Monge map for Gaussians are known in closed form.

To limit content loss, consider the McCann interpolates $(1-t)\text{Id} + tT$ instead. To interpolate between styles, use Wasserstein barycenters instead of linear interpolation.

***From entropy-regularized optimal transport
to Sinkhorn divergences***
A. Genevay (NeurIPS 2019)

There are three ways of computing “distances” between distributions:

- Csiszár ϕ -divergence, e.g., the Kullback-Leibler divergence;
- Maximum mean discrepancies,

$$\text{MMD}(\alpha, \beta) = \sup_{\|f\|_{\mathcal{H}} \leq 1} \left| \mathbb{E}_{x \sim \alpha}[f(X)] - \mathbb{E}_{y \sim \beta}[f(Y)] \right|$$

for some RKHS \mathcal{H} ;

- Optimal transport, which is slower to compute, but provides better gradients.

With an entropic regularization

$$H(\pi|\alpha \otimes \beta) = \int \log \frac{d\pi(x, y)}{d\alpha(x)d\beta(y)} d\pi(x, y),$$

the dual is an unconstrained optimization problem and the Sinkhorn algorithm solves it for discrete distributions. But this regularization assumes the data is observed with noise, and tries to remove it: you get



when you would prefer



The regularized Wasserstein distance is not a distance: $d_{W,\varepsilon}(\alpha, \alpha) \neq 0$. Instead, consider the *Sinkhorn divergence*,

$$\text{SD}_{\varepsilon}(\alpha, \beta) = d_{W,\varepsilon}(\alpha, \beta) - \frac{1}{2}d_{W,\varepsilon}(\alpha, \alpha) - \frac{1}{2}d_{W,\varepsilon}(\beta, \beta).$$

For $\varepsilon \rightarrow \infty$, it converges to a MMD.

***How does mini-batching
affect Curvature information
for second order deep learning optimization?***
D. Granziol et al. (NeurIPS 2019)

$\text{Spec}(A + \text{noise})$ is not an unbiased estimator of $\text{Spec} A$. In particular, the largest eigenvalue is perturbed upwards. Random matrix theory (RMT) can correct this bias.

Acceleration through spectral modeling
F. Pedregosa and S. Scieur (NeurIPS 2019)

Accelerated gradient methods only use the largest (and smallest) eigenvalue(s) of the Hessian, ignoring the rest of the spectrum: model the spectral distribution as a *Marcenko-Pastur* distribution (fitted using the largest eigenvalue and the trace) and use it to find the optimal first-order method. In special cases, this is equivalent to the *heavyball* method.

***The importance of better models
in stochastic optimization***
H. Asi and J Duchi (NeurIPS 2019)

SGD is very sensitive to step size and can diverge. **(Approximate) proximal optimization** (aProx) minimizes an approximation of the objective with a penalty to keep the new point close to the previous. Approximations include:

- The function itself (*i.e.*, no approximation, just a penalty);
- A linear approximation;
- A linear approximation floored at zero (it is often known that $f \geq 0$): this prevents divergence and can be implemented by replacing $x \leftarrow x - \alpha g$ with

$$x \leftarrow x = \text{Min}\left\{\alpha, \frac{f(x)}{\|g\|^2}\right\}g.$$

Beyond SGD: in search of a preconditioner
E. Hazan (NeurIPS 2019)

Preconditioning makes the loss surface more isotropic.

LiSSA (linear time second order stochastic optimization) writes the Hessian inverse as an infinite series (assuming the eigenvalues are below 1 in absolute value) and samples from it; to estimate the powers, use iid samples (one sample for each, *i.e.*, rank-1 matrices).

$$H^{-1} = \sum_{k \geq 0} (I - H)^k$$

$$H^{-1} \approx (I - H)^k p \cdot p_k^{-1}$$

$$(I - H)^k \approx \prod_{1 \leq i \leq k} (I - H_i)$$

For non-convex functions, use cubic regularization (FastCubic). For neural nets, there is no improvement on SGD.

Replace AdaGrad $x \leftarrow x - \text{diag}(\sum g_t g_t')^{-1/2} g$ (you can take the full matrix instead of the diagonal) with GGT , $x \leftarrow x - (G_t G_t')^{-1/2} g_t$, where $G_t = [g_t | \beta g_{t-1} | \beta^2 g_{t-2} | \dots]$. It can be efficiently implemented using $(GG')^{-1/2} x = G(G'G)^{-3/2} x$, since $G'G$ is smaller than GG' . Plot the eigenvalues of GG' against the epoch.

AdaGrad keeps two gradients, Adam three. To reduce memory usage, only keep a “low-rank approximation” of the gradients (first reshape them into matrices or tensors): *extreme tensoring*.

Symmetric and multi-secant quasi-Newton methods
D. Scieur et al. (NeurIPS 2019)

Quasi-Newton methods approximate the Hessian using the gradient and the secant equation $f''(x)^{-1}(f'y - f'x) \approx y - x$. Let Δx and ΔG be the matrices containing the first differences of successive estimates and gradients, and H be an estimator of $(f'')^{-1}$. The secant equation becomes $H\Delta G = \Delta X$. Quasi-Newton methods find a matrix H , close (for some norm), to some reference matrix H_0 , such that $H\Delta G = \Delta X$. For instance, LBFG uses $H_0 = H_k$, $N = 1$ seconds, $d(H, H_0) = \|H - H_0\|_{\nabla^2 f}$. For $N > 1$, the secant equation has no solution in general: use an approximation instead (it reduces to a Procrustes problem).

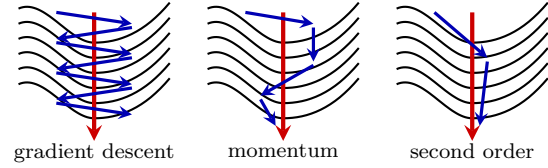
Stochastic Newton and cubic Newton methods with simple local linear quadratic rates
D. Koralev et al. (NeurIPS 2019)

Cubic regularization generalizes Newton’s method

$$\langle f'x_k, x - x_k \rangle + \frac{1}{2} \|x - x_k\|_{f''(x_k)}^2 + \frac{M}{6} \|x - x_k\|^3.$$

Subsampled methods estimate the gradient and the Hessian on different (independent) subsets; use a very large minibatch for the Hessian.

K-FAC: extensions, improvements and applications
J. Martens (NeurIPS 2019)



Second-order methods should be faster, but the curvature changes very quickly: the second-order approximation is only valid locally – use a *trust region*, either as a hard constraint or, equivalently, as an L^2 penalty.

One can replace the Hessian with GGN, Fisher information or (not recommended) empirical Fisher information.

The **generalized Gauss–Newton** method approximates the Hessian of

$$h(\theta) = \frac{1}{m} \sum_{i=0}^m L(y_i, f(x_i, \theta))$$

as $G = \frac{1}{m} \sum J_i' H_i J_i$ where $J_i = \nabla_{\theta} f$, $H_i = \nabla_{zz} L$. This is the Hessian of h with f replaced with its first-order approximation around θ . The square norm loss, $L(y, z) = \frac{1}{2} \|y - z\|^2$ gives $H = I$ and $G = \frac{1}{m} \sum J_i' J_i$ (Gauss–Newton). When z is the natural parameter of an exponential family, G is the Fisher information and $G^{-1} \nabla h$ the natural gradient. GGN works better than the Hessian in practice.

The Hessian is too large, but many approximations are available: diagonal (Adam), block-diagonal (TONGA), low-rank + diagonal (L-BFGS), Krylov (Newton-CG, HF).

The **Kronecker-factorized approximate curvature** (K-FAC) exploits the structure in layers of the neural net. For a single linear layer, $a \mapsto s = Wa$, and $L(y, f(x, \theta)) = -\log p(y|x, \theta)$, $\nabla_{\theta} L = ga'$, where $g = \nabla_s L$ and the Fisher information is $F = \text{Cov}(\text{vec } \nabla_w, \text{vec } \nabla_w) = \dots - E[aa'] \otimes E[gg'] = A \otimes G$. The Kronecker product allows efficient computations: $(B \otimes C)^{-1} = B^{-1} \otimes C^{-1}$, $(B \otimes C) \text{vec } X = \text{vec}(CXB')$.

Shared weights, in CNNs or RNNs, complicate things.

K-FAC works better than SGD for large batch sizes. TensorFlow implementation available.

HAWKV2: Hessian-aware trace-weighted quantization of neural networks
A. Gholami (NeurIPS 2019)

To choose how many bits to keep for each layer, use the trace of the Hessian, computed as

$$\text{tr } A = \mathbb{E}_{z \sim \text{Rademacher}} [z' A z].$$

Implementation: **PyHessian**.

New methods for regularization path optimization via differential optimization
P. Grigas and H. Liu (NeurIPS 2019)

The first order condition defining the regularization path, $\nabla_x F_\lambda(x(\lambda)) = 0$ leads to an ODE, which can be solved with the Euler scheme (or a better one); it involves the Hessian, but computations can be avoided using the conjugate gradient (CG).

Foundations of causal inference: challenges and opportunities of big data
N. Kiyavash (NeurIPS 2019)

Granger causality can be extended to point processes, using *directed information* as a loss function; the resulting *directed information graph* (DIG) is a Bayesian network; it can also, more easily, be estimated from a model of the point process, e.g., a *multivariate Hawkes process*. Applications include neural spike trains, twitter activity, econometric modeling of systemic risk, hyperlink creation between (one million) websites (showing clusters of merchant and review sites).

Insider threat detection via hierarchical neural temporal point processes
X. Wu (NeurIPS 2019)

Detect fraudulent sessions (from login time, email activity, web activity, device access, opening/copying/creating/deleting files) by combining RNNs (LSTM) and masked temporal point processes (MTPP), using encoder-decoders, predicting activity type and time (intra-session) and session time (inter-session).

Datasets: CERT insider threat, Wikipedia vandalism.

Temporal point processes: web, social media, and networking systems
N. Ganguly (NeurIPS 2019)

Hawkes-RNN models can be used to study:

- Temporal and information influence of social media users; latent opinion inference;
- Tweet-hashtag (reinforcement) and hashtag-hashtag interactions;
- Spikes in network traffic.

Temporal logic point processes
S. Li et al. (NeurIPS 2019)

Add prior knowledge (“event A happens before B”) to temporal point processes (TPP) using *temporal logic* soft constraints (with learned weights).

The graph Hawkes network for reasoning on temporal knowledge graphs
Z. Han et al. (NeurIPS 2019)

A *temporal knowledge graph* stores (S,V,O,timestamp) tuples. In a *neural Hawkes process*, the intensity of events of type k is $\lambda_k = f(w'_k h_t)$. However, there are

too many event types, and some are concurrent. Learn vector representations of entities and predicates; respond to queries of the form (S,V,?,t).

Data: political events, news.

Deep point process destructors
D. Inouye (NeurIPS 2019)

Destructive learning identifies patterns in the data, removes them in an invertible way, and starts again. For point processes, use time warping. A point process destructor is a time transformation that maps the point process studied to the unit process $\lambda \equiv 1$.

For deep point processes destructors, e.g. if $D = D_2 \circ D_1$,

$$\lambda(t) = \frac{dD}{dt} = \lambda_2(D_1(t))\lambda_1(t).$$

More generally, a *deep point process* replaces the mixture $\lambda_t = \sum_i \lambda_i(t)$ with $\lambda(t) = \prod_i \lambda_i(t^{(i-1)})$, $t^{(0)} = t$, $t^{(i)} = D_i(t^{(i-1)})$ (each D_i can be a weak model).

Bandits, estimation and hypothesis testing
S. Athey (NeurIPS 2019)

Bandits (aka *adaptive experiments* in statistics) are more data-efficient than traditional experiments. Bandits create unbalanced data (they are not iid); naively estimating the value of an arm is potentially biased and non-Gaussian. *Inverse propensity weighting* addresses those problems but increases variance; there are other weighting schemes.

Preserving causal constraints in counterfactual explanations for machine learning classifiers
(NeurIPS 2019)

Counterfactual explanations should be actionable (you cannot decrease your age or education level) and satisfy some constraints (you cannot be under 20 and have a PhD): use a structural causal model.

EconML: a machine learning library for estimating heterogeneous treatment effects
(NeurIPS 2019)

Includes: DeepIV, DoubleML, causal forests, etc.

Adaptive TRPO: convergence and faster rates for regularized MDPs
L. Shani et al. (NeurIPS 2019)

Mirror descent is the trust-region optimization algorithm

$$x_{k+1} \leftarrow \underset{x}{\operatorname{Argmin}} \langle \nabla f(x_k), x - x_k \rangle + \alpha_k B_\omega(x, x_k)$$

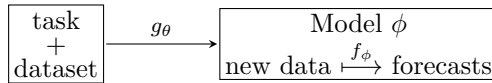
where the first term is a linear approximation of the objective and the second term is a *Bregman divergence*

$$B_\omega(x, y) = \omega(x) - \omega(y) - \langle \nabla \omega(y), x - y \rangle.$$

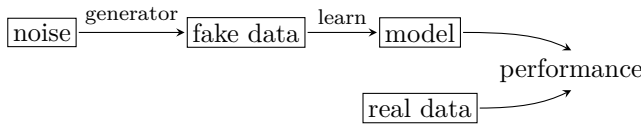
If ω is the Euclidean norm, B_ω is the Euclidean distance and mirror descent is projected gradient descent.

If ω is the negentropy, B_ω is the Kullback-Leibler divergence, and mirror descent is exponentiated gradient descent.

How metalearning could help us accomplish our grandest AI ambitions, and early, exotic steps in that direction (NeurIPS 2019)



Generative teaching networks (GTN) generate data to learn on. Backpropagate to train the generator, but use *weight normalization*. Use different model architectures.



In RNNs, do not only store information in the activations, but also in the weights, using *Hebbian learning*: “neurons that fire together wire together”.

$$w_{ij}^{t+1} = w_{ij}^t + \eta x_i^t x_j^t.$$

Train w and α (“differentiable plasticity”) with SGD:

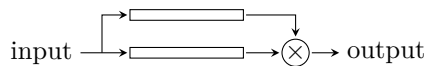
$$y_i = \tanh \sum_j (w_{ij} + \alpha_{ij} H_{ij}^t) y_j$$

$$H_{ij}^{t+1} = \eta y_i y_j + (1 - \eta) H_{ij}^t.$$

In *modulated Hebbian learning*, the Hebbian trace H_{ij} is only updated if some neuron M fires:

$$H_{ij}^{t+1} = H_{ij}^t + M_t y_i^{t-1} y_j^t.$$

To prevent *catastrophic forgetting*, use a mask to select which part of the network to use for the current task.



One-ended algorithms (natural selection, human culture) generate their own problems and solve them. The “paired open-ended trailblazer” (POET) generates environments of varying difficulties (e.g., landscape, for a robot to navigate in); trains an agent on each of them in parallel; re-evaluate the agents on all models and copy them accordingly (this works better than linearly interpolating between easy and difficult environments – curriculum learning is hard, and a linear curriculum is rarely optimal).

Better model-based RL through meta-RL P. Abbeel (NeurIPS 2019)

Use *domain randomization*: simulated environments do not look like the real world, but they may be as different between them as they are different from the real world – instead of using just one simulated environment, use many, and measure the performance on unseen environments.

Overfitting plagues model-based RL much more than supervised learning: the agent would try to exploit regions of the state space where the model is imprecise but promising. Use domain randomization, e.g., with an ensemble of models.

Train asynchronously.

To speed up meta-RL: train separate RL agents for all the tasks, then train a global agent to match those local agents (imitation learning)

Abstraction and meta-RL D. Abel (NeurIPS 2019)

State abstraction tries to make reinforcement learning easier by replacing the state space with a smaller, discrete set, with no drop in the quality of the solution learned.

Action abstraction replaces the action space with a larger one, by adding *options*, i.e., long-term actions (defined by a start condition, an end condition, and a policy). Finding options that minimize planning time is NP-hard, and hard to approximate (at least in the worst case).

Scalable metalearning R. Hadsell (NeurIPS 2019)

WarpGrad learns to precondition the gradient – it learns the geometry of the space while optimizing the model.

Vanilla transformers do not work well in RL: add (GRU-like) gating.

Metalearning with warped gradient descent S. Flennerhag (NeurIPS 2019)

Learn how to warp the loss surface into something nice; this is equivalent to learning a gradient preconditioner.

Fairness assessment for AI in the finance industry (NeurIPS 2019)

AIF360 is a Python library for fairness measures and mitigation algorithms.

Forecasting firm material event sequences from SEC 8k reports (NeurIPS 2019)

Seq2seq transformer model to map past events to future events; the attention maps provide explanations.

Explainable small business credit scoring
(NeurIPS 2019)

Accounting software companies have access to detailed accounting data and can use it to decide whether to grant a loan (xgboost, monotonicity constraints, Shapley values).

***Adversarial learning of “deep fakes”
in accounting***
(NeurIPS 2019)

Adversarial autoencoders impose a structure on the latent representation, e.g., a mixture of clearly separated Gaussians, to disentangle.

Auditors use ERP data: a fraudulent company could use an adversarial attack to hide anomalous transactions.

Imperceptible attacks on tabular data
(NeurIPS 2019)

Tabular data is also susceptible to adversarial attacks. Since not all features are equally important (you can lie more about your number of pets than about your salary), define a notion of “perceptability” and minimize it to mount your attack (LowProFool, DeepFool, FGSM).

Understanding equilibria in multiagent systems
M. Woolridge (NeurIPS 2019)

While Siri does not (yet) talk to Siri, *multiagent systems* are already there, e.g., with algorithmic trading – but those systems are unstable and can lead to flash crashes.

1. To understand this instability, consider it as a bug, and use *model checking*: build the state transition graph of the system and use *temporal logic* (LTL, an extension of propositional logic with temporal operators)

$\Diamond\phi$	eventually
$\Box\phi$	always
$\phi\mathcal{U}\psi$	until
$\Box\Diamond\phi$	infinitely often
$\Diamond\Box\phi$	always after some point

to formulate and check the desired properties of the system, in particular reachability $\Diamond\phi$ and invariance $\Box\phi$.

2. This is too restrictive: an undesirable state could be reachable only through a sequence of irrational decisions. Use the *Reactive Modules* language to describe the agents and their goals, and check reachability under Nash equilibrium.

3. This approach only works on small systems: for large systems, use *agent-based models*, calibrated on real data, and simulations.

The flash crash prevention measures (stop trading if the prices drop too much) seem ineffective and even

counterproductive: if investors need to sell ad cannot, they will find something else to sell – the flash crash is contagious.

Putting ethical AI to the vote
A. Procaccia (NeurIPS 2019)

When there are several stakeholders for a given problem (e.g., food donations allocation), if the problem is recurring, build a model for each stakeholder and aggregate their preferences (*social choice*). Since the output of the model is noisy, the aggregation mechanism, should be robust to noise: Borda count is robust, pairwise-majority consistent (PMC) methods are not. [This is currently used as a decision-support tool; it is more efficient than humans in terms of diversity and cost.]

Other applications of this *virtual democracy* (the models vote, or the humans) include ethical decisions for self-driving cars (aggregating the ethical choices of one million Americans in a fraction of a second), or ESG investment (when several people, e.g., a family, want to invest together, but do not have the same views on ESG).

***Gaussian process behaviour
in wide deep neural networks***
A.G.G. Matthews (NeurIPS 2019)

As the width of a 1-layer neural net tends to infinity, it converges to a Gaussian process (natural tangent kernel). This is still the case for k -layer neural nets, if deeper layers grow faster.

The natural tangent kernel
T.G.J. Rudner et al. (NeurIPS 2019)

In continuous time, gradient descent is just an ODE, $\dot{\theta} = -\eta\nabla_{\theta}\mathcal{L}(\theta)$; it also gives us the dynamics of the output of the neural net during gradient descent: a diffusion, for the *neural tangent kernel*

$$\Theta(x, y) = \nabla_{\theta}f(\theta)\nabla_{\theta}f(\theta)'.$$

For natural gradient descent, this becomes

$$\Theta^{\text{nat}}(x, y) = \nabla_{\theta}f(\theta)F(\theta)^{-1}\nabla_{\theta}f(\theta)',$$

where $F(\theta)$ is the Fisher information matrix. Linearize those dynamics and solve the resulting ODE, instead of SGD to train the network.

Deep ensembles: a loss landscape perspective
S. Fort et al. (NeurIPS 2019)

Ensembles work better than Bayesian neural nets: ensemble can cover several modes; Bayesian approaches struggle to (since the modes are connected, they should be able to).

Average the weights around each mode, and the predictions across modes.

MIM Mutual information machine
M. Livne et al. (NeurIPS 2019)

Replace the VAE loss

$$\ell(\theta) = \text{KL}(q_\theta(z|x)p(x) || p_\theta(x|z)p(z))$$

where x is the observed variable and z the latent representation, with the more symmetric Jensen-Shannon divergence, and add a regularizer

$$H(q_\theta(z|x)p(x)) + H(p_\theta(x|z)p(z))$$

(entropy of the two joint distributions on (x, z)) to help maximize the mutual information between x and z

$$H(x, z) = H(x) + H(z) - I(x; z).$$

Pitfalls of in-domain uncertainty estimation and ensembling in deep learning
(NeurIPS 2019)

Metrics used for uncertainty estimation (log-likelihood, Brier score, calibration error, AUC) have pitfalls, but can be fixed.

Neural tangents: fast and easy infinite neural networks in Python
R. Novak et al. (NeurIPS 2019)

Infinite-width neural nets (Gaussian processes) in JAX/stax.

Using loss surface geometry for practical Bayesian deep learning
A.G. Wilson (NeurIPS 2019)

<http://losslandscape.com/>

A simple baseline for Bayesian uncertainty in deep learning
W. Maddox et al. (NeurIPS 2019)

SWAG fits SGD iterates with a low-rank + diagonal Gaussian distribution.

Can you trust your model's uncertainty?
J. Snoek et al. (NeurIPS 2019)

Measures of uncertainty include the expected calibration error (ECE), which can be gamed, and *proper scoring rules* such as the negative log-likelihood, which emphasizes tails too much, or the Brier score.

Methods include:

- Post hoc calibration, with temperature scaling;
- Ensembling;
- Monte Carlo dropout;
- Stochastic variational inference;
- Bayesian only in the last layer.

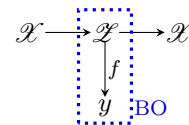
Under distribution shift, post hoc calibration is the worst, ensembling (with small ensembles: 5) is the best.

Dropout as a Bayesian approximation: representing model uncertainty in deep learning
Y. Gal and Z. Gharhamani (2015)

Dropout performs approximate Bayesian inference.

High-dimensional Bayesian optimization using low-dimensional feature spaces
R. Moriconi et al. (NeurIPS 2019)

Bayesian optimization in high dimension ($d \geq 20$) can use a random projection, an additive decomposition of the objective function, or a nonlinear embedding, learned with a VAE, progressively, during optimization.



Function space prior in Bayesian deep learning
R. Grosse (NeurIPS 2019)

The *automatic statistician* searches for compositional GP kernels describing the data (and can output an English text describing the model): having a better prior makes it better at extrapolation.

The *neural kernel network* (NKN) is a neural network computing $(x, y) \mapsto k(x, y)$; each layer combines primitive kernels (or kernels from previous layers), using additions and multiplications (good for extrapolation and exploration).

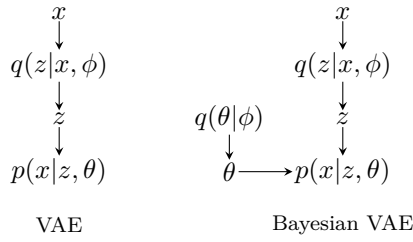
Functional variational BNNs specify a prior on the stochastic process, not on the weights; variational inference can be done on function (not weight) space. [GPs are currently still preferable.]

Try depth instead of weight correlation
S. Farquhar et al. (NeurIPS 2019)

The widespread belief that the approximate posterior for Bayesian neural nets needs correlations between weights only holds for shallow networks. Even for linear networks, $n \geq 3$ layers gives a full correlation matrix for the outputs.

Deep generative models for genetic variation and drug design
D. Marks (NeurIPS 2019)

The alignment and evolution of biological sequences (mutation, insertion, deletion) can be modeled with Bayesian VAEs or Bayesian seq2seq models, and be used to sample from possible future flu viruses to make vaccines.



**Measure-valued derivatives
for approximate Bayesian inference**
M. Rosca et al. (NeurIPS 2019)

Besides the REINFORCE (score function) and reparametrization estimators of the gradient of an expectation

$$\begin{aligned}\frac{d}{d\theta} \mathbb{E}_{x \sim p_\theta} f(x) &= \frac{d}{d\theta} \int f(x) p_\theta(x) dx \\ &= \int f(x) \frac{dp_\theta}{d\theta} dx \\ &= \int f(x) \frac{dp_\theta}{d\theta} \frac{p_\theta(x)}{p_\theta(x)} dx \\ &= \int f(x) \frac{d \log p_\theta}{d\theta} p_\theta(x) dx \\ &= \mathbb{E}_{x \sim p_\theta} [f(x) \nabla_\theta \log p_\theta(x)] \\ \frac{d}{d\theta} \mathbb{E}_{x \sim p_\theta} f(x) &= \frac{d}{d\theta} \mathbb{E}_{x \sim p} f(g_\theta(x)) \\ &\text{if } z \sim p \implies g_\theta(z) \sim p_\theta\end{aligned}$$

the *measure-valued derivative* is less known

$$\begin{aligned}\frac{d}{d\theta} \mathbb{E}_{x \sim p_\theta} f(x) &= \int f(x) \frac{dp_\theta}{d\theta} dx \\ &= c_\theta \left(\mathbb{E}_{x \sim p_\theta^+} f(x) - \mathbb{E}_{x \sim p_\theta^-} f(x) \right)\end{aligned}$$

[when sampling from p_θ^+ and p_θ^- , use the same random numbers, to lower variance] where $dp_\theta/d\theta = c_\theta(p_\theta^+ - p_\theta^-)$ is the decomposition of the signed measure $dp_\theta/d\theta$ into difference of probability measures, up to a multiplicative factor (e.g., *Hahn-Jordan decomposition*) [the authors provide a table of such decompositions]. For instance, for the Gaussian distribution,

$$\frac{d}{d\mu} \phi_{\mu,1}(x) = \frac{1}{\sqrt{2\pi}} (W_{2,.5}(x - \mu) - W_{2,.5}(\mu - x))$$

where W is the Weibul distribution.

This requires no strong assumptions (it works with discrete distributions and discontinuous f), has low variance, but requires $2|\theta|$ evaluations of f (in terms of clock time, the score function with baseline is faster for $|\theta| \geq 100$).

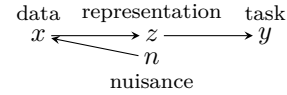
**Monte Carlo gradient estimation
in machine learning**
S. Mohamed et al. (2019)

Survey article: why we need the gradient

$$\nabla_\theta \mathbb{E}_{x \sim p_\theta} [f_\theta(X)]$$

and three ways of computing it: score function (REINFORCE), reparametrization (“pathwise”) and measure-valued.

**Information in the weights and emergent
properties of deep neural networks**
S. Soatto and A. Achille (NeurIPS 2019)



Supervised learning looks for a representation z , as small as possible, *i.e.*, minimizing $I(x; z)$, containing as much information on the task as the input, *i.e.*, $I(z; y) = I(x; y)$. The Lagrangian of the constrained optimization problem is

$$\mathcal{L}(p(z|x)) = \underbrace{H(y|z)}_{\text{crossentropy}} + \beta \cdot \underbrace{I(z; x)}_{\text{regularizer}}.$$

Troublingly, setting z to the index of the image in the training set is close to optimal (16 bits on CIFAR): the Lagrangian should be minimized, not on the training data, but on the data we do not have yet.

Information is also stored in the weights. However, since the weights are real, the information is infinite, and most if it is irrelevant: the relevant information can be measured by how much noise can be added to the weights without changing the result (or the drop in accuracy as a function of noise). More precisely,

$$\begin{aligned}\text{Minimize} & \quad \text{KL}(q(w|\text{data})||p(w)) \\ \text{Such that} & \quad \mathbb{E}_{w \sim q(w|\text{data})} [\mathcal{L}_{\text{data}}(w)] \leq t\end{aligned}$$

with Lagrangian

$$\mathcal{L} = \underbrace{\mathbb{E}_{w \sim q(w|\text{data})} [\mathcal{L}_{\text{data}}(w)]}_{\text{expected loss}} + \beta \cdot \underbrace{\text{KL}(q(w|\text{data})||p(w))}_{\text{information in the weights}}$$

SGD minimizes the Fisher information of the weights. The distance between two tasks (for transfer learning) can be defined as

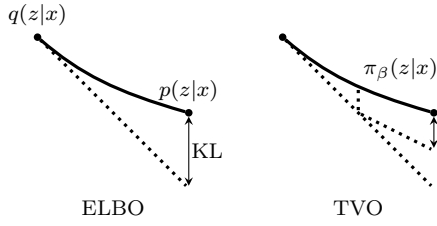
$$d(\mathcal{D}_1 \rightarrow \mathcal{D}_2) = \underbrace{I(\mathcal{D}_1 \mathcal{D}_2; w)}_{\text{complexity of learning together}} - \underbrace{I(\mathcal{D}_1; w)}_{\text{complexity of learning one}}$$

Alternatively, one can represent a task with the diagonal of its Fisher information matrix (task2vec),

$$d(\mathcal{D}_1, \mathcal{D}_2) = \cos(F_1, F_2).$$

**Understanding
thermodynamic variational inference**
R. Brekelmans et al. (NeurIPS 2019)

The *thermodynamic variational objective* gives a tighter alternative to the ELBO.



**GAIT:
a geometric approach to information theory**
J. Gallego et al. (NeurIPS 2019)

Shannon entropy considers that all symbols are different in the same way. Information theory can be extended to account for the geometry of the space – like the Wasserstein distance, but in closed form. Consider a *similarity space* (\mathcal{X}, κ) ,

$$\begin{aligned} \forall x, y \quad 0 &\leq \kappa(x, y) \leq 1 \\ \forall x \quad \kappa(x, x) &= 1 \end{aligned}$$

and a probability distribution P on \mathcal{X} .

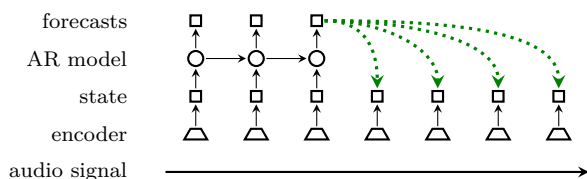
$$\begin{aligned} KP(x) &= \mathbb{E}_{y \sim p} [\kappa(x, y)] && \text{ordinariness of } x \\ 1/KP(x) &&& \text{rarity of } x \\ H_1^k[p] &= \mathbb{E}_{x \sim p} [-\log KP(x)]. \end{aligned}$$

It is conjectured that H_1^k is concave (under reasonable conditions): we can use $-H_1^k$ to define a similarity-sensitive Bregman divergence.

$$D^k(P\|Q) = 1 + \mathbb{E}_{x \sim P} \left[\log \frac{KP(x)}{KQ(x)} \right] - \mathbb{E}_{x \sim Q} \left[\log \frac{KP(x)}{KQ(x)} \right]$$

Contrastive predictive coding
A. van den Oord (NeurIPS 2019)

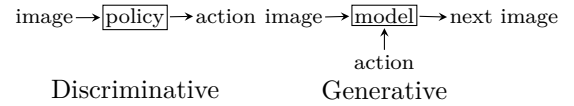
Contrastive predictive coding (InfoNCE, noise contrasting encoding) learns to distinguish pairs with the same label, e.g., $(\text{dog}_1, \text{dog}_2)$ from pairs with different labels, e.g., $(\text{dog}_1, \text{cat}_2)$. Labels are not strictly necessary to learn a useful representation: just assume that nearby patches (of the sound signal, etc.) are from the same class (phoneme, speaker, etc.) while other matches from the dataset are unrelated.



**Perception as generative reasoning:
structure, causality probability**
(NeurIPS 2019)

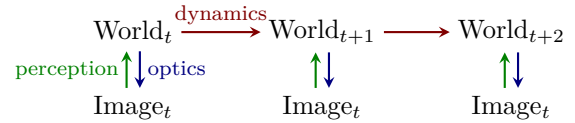
$$\begin{aligned} \text{Discriminative model} & \quad p(\text{label}|\text{image}) \\ \text{Generative model} & \quad p(\text{image}|\text{label}) \times p(\text{label}) \end{aligned}$$

In the generative model, the first factor can include physics engines and renderers, the second can add structure, e.g., $p(\text{label}) = p(\text{label}|z)p(z)$.



**Perception and action
from generative models of physics**
K. Smith (NeurIPS 2019)

Perception is inverse optics informed by dynamics.



It can be used to keep track of 3D objects under occlusion.

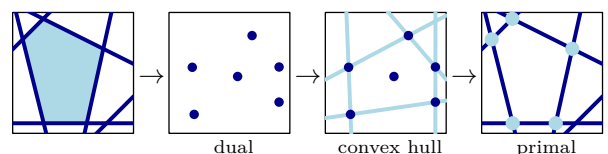
CvxNet: learnable convex decomposition
B. Deng et al. (NeurIPS 2019)

Vision is inverse graphics: a complex shape can be described using simple primitives, e.g., boxes, ellipsoids or, more generally, convex shapes, defined implicitly as intersections of half-spaces

$$\Phi(x) = \text{LogSumExp}(\delta(Nx + d)) \quad (\text{softmax}).$$

A (union) of convex shapes can be learnt with a (multi-headed) auto-encoder.

To convert the implicit representation into an explicit one (extreme points), compute the convex hull in the dual (cf. F-Rep, *function representation*)



Variational graph convolutional networks E.V. Borilla (NeurIPS 2019)

Graph convolutional networks (GCN) are not robust to noise in the graph structure:

- Add a prior on the graph structure;
- Use a *low rank* parametrixation of the variational posterior

$$q(A) = \prod_{ij} q(A_{ij})$$

$$q_{ij} = \text{Bernoulli}(A_{ij}, \rho_{ij})$$

ρ : low rank

- Relax the discrete Bernoulli distribution into a “binary concrete” distribution.

Representation and generation of molecular graphs W. Jin (NeurIPS 2019)

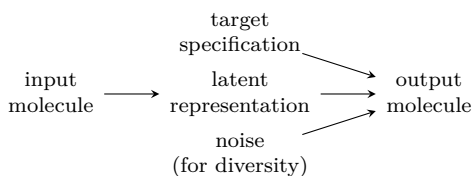
(Permutation-invariant) graph neural nets (GNN) are not expressive enough: they cannot predict girth, circumference, diameter, total number of cycles, etc. In practice, we add cycle-related features, e.g., whether an atom is in an aromatic ring.

View molecules at multiple levels, using a dictionary of structures,



which define a coarser graph, and use message passing, e.g., to forecast molecule properties (data from the ZINC database).

For *de novo molecule optimization*, modify an existing molecule into a new one, usable as a drug, using an encoder-decoder network, and incrementally construct the hierarchical graph of the output. Use the EM algorithm to satisfy the specifications: the E step generates candidates and reweights them, the M step maximizes the weighted log probability.



Proposal for an open graph benchmark J. Leskovec (NeurIPS 2019)

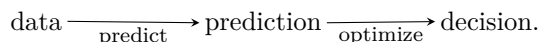
PyTorch code available (dgl, pytorch_geometric): <http://ogb.stanford.edu/>.

Graph representation learning for optimization on graphs B. Dilkina (NeurIPS 2019)

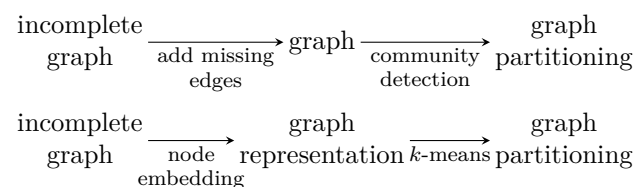
1. Machine learning components can be used inside optimization algorithms, in particular when we have to

repeatedly solve the same problem with new data. For instance, one can use reinforcement learning to learn the scoring function (Q -value function) of a greedy algorithm to solve combinatorial optimization problems on graphs such as minimal cover (the degree is often used as a score, but one can do better), maximum cut or TSP, from the features of the graph – the graph representation (features) is learnt by structure2vec.

2. Two-stage processes are often suboptimal:



A pure end-to-end approach is possible, but combinatorial optimization complicates things. A simpler approach is to find a related problem, easier to solve, and learn a mapping from the original problem to the simpler one.



(For the backward pass, only unroll one of the k -means iterations.)

Deep graph library Z. Zhang (NeurIPS 2019)

PyTorch library for (message-passing) neural nets. Also check `pytorch_geometric`.

Graph networks for learning physics P. Battaglia (NeurIPS 2019)

Graph networks are a formalism for graph neural networks based on *learned message passing*.

$$v_i, v_j : \text{node features}$$

$$e_{ij} : \text{edge features}$$

$$\phi : \text{neural networks}$$

$$e'_{ij} = \phi(e_{ij}, v_i, v_j) \quad \text{edge function}$$

$$e_i = \sum_j e_{ji} \quad \text{aggregation}$$

$$v'_i = \phi(e_i, v_i) \quad \text{node function}$$

They can learn physical systems from observations, e.g., n -body problems, balls bouncing in a box, springs; the learned model generalizes to larger systems.

Once they have learned the physical laws, one can replace the Euler integrator (computing the state of the system at the next time) with a better one.

Code: `graph_net`; also check `dgl`, `pytorch_geometric`.

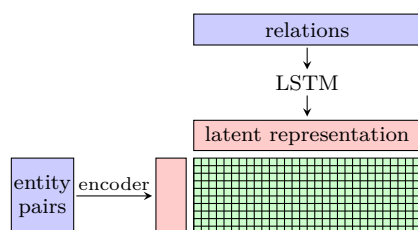
Learning DAGs and trees with box embeddings and hyperbolic embeddings

A. McCallum (NeurIPS 2019)

Knowledge can be represented as raw (indexed) text, a trained deep learning model (BERT – type 1 reasoning) or a knowledge base (KB – type 2 reasoning).

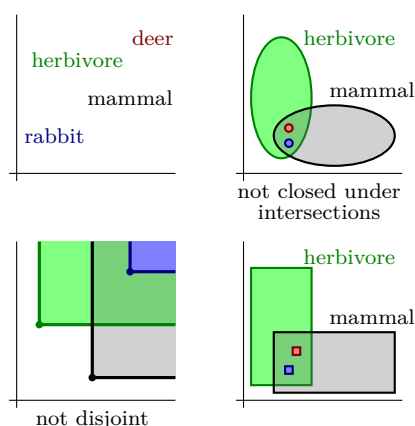
From text, extract entities and relations; put them in an entity pair \times relation matrix, and use classical matrix completion, or “multi-hop reasoning” to infer the missing relations.

Instead of using actual entity pairs and relations as rows and columns, use a latent representation: it will accommodate entities and relations not seen during training.

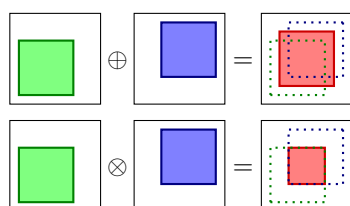


Add types to those entities, with a hierarchical structure (e.g., president \Rightarrow human), learned with a *bilinear model* ($c_1^T A c_2 > 0$ iff $1 \Rightarrow 2$, i.e., 1 is_a 2, i.e., $1 \subset 2$).

To add probabilities to those hierarchical relations (e.g., $P[\text{president} \Rightarrow \text{writer}] = .8$), do not represent concepts as points in a vector embedding, but as Gaussian densities, cones or boxes.



Those boxes can be transformed with a neural net once you define addition and multiplication of boxes.



To learn those hierarchies from scratch, embed concepts in *Poincaré space*: this gives a continuous representation of trees.

Representation learning and fairness

S. Koyejo (NeurIPS 2019)

Fairness can be ensured by

- Pre-processing: representation learning, feature adjustment, metric learning;
- In-processing: learning with fairness penalties;
- Post-processing: adjustments.

For the preprocessing approach, clearly separate the roles:

- The *data regulator* defines the fairness criteria and audits the sanitized data and the final model;
- The *data producer* turns the raw data into a fair representation;
- The *data user* uses the sanitized data to fit a model.

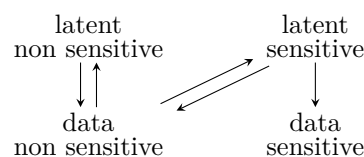
There are many, often incompatible, fairness criteria. *Individual fairness* requires that “similar” individuals be treated “similarly” – this requires a metric. *Group fairness* asks that some statistics be equalized across predefined groups, e.g., *statistical parity* (TP+FP), *equalized odds* (TP, FP), *equality of opportunity* (TP). To choose the statistics to equalize, present the end user with several confusion matrices and ask her which ones she prefers (metric elicitation). Group fairness does not imply individual fairness; fairness for attributes A and B does not imply joint fairness for (A, B) .

For individual fairness, the regulator can provide sets of examples which should be treated in the same way, and the data producer trains a model using

$$x_1 \sim x_2, x_1 \not\sim x_3 \implies \|x_1 - x_2\| \leq \|x_1 - x_3\|.$$

For group fairness one can replace the observations with prototypes ensuring statistical parity.

Disentanglement trains a VAE which separates the sensitive attributes from the non-sensitive ones.



There is a trade-off between accuracy and fairness.

Pay attention to inappropriate data, e.g., feedback effects.

Synthetic control

A. Abadie et al. (NeurIPS 2019)

To estimate the effects of an intervention from *observational* data, in particular when the intervention is applied to large units (cities, countries) – this is sometimes needed because of fairness considerations or possible interactions – match each treated unit, not with a single control, but with a weighted average of controls. The synthetic control is computed by constrained optimization, matching the features of the unit and the pre-intervention values.

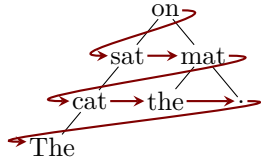
It may be useful to denoise the time series first, e.g., with a low-rank approximation, minimizing $\|\hat{Y} - Y\|_{2,\infty}^2$ rather than $\|\hat{Y} - Y\|_F^2$. This can be generalized to multiple interventions (“what if unit i had received intervention j instead of i ?”), with a low-rank tensor decomposition.

Examples include: German GDP after reunification, smoking regulations in California, UK GDP after Brexit, cricket results if it had not rained.

**Machine learning
for computational biology and health**
A. Goldenberg and B. Engelhardt (2019)

**Imitation learning
and natural language generation**
K. Cho and H. Daumé (NeurIPS 2019)

One can generate text in a non-monotonic order.



Deep learning with Bayesian principles
M.E. Khan (NeurIPS 2019)

Deep learning scales well with data and model complexity; Bayesian learning estimates uncertainty and can be incrementally updated. Bayesian updates are global; deep learning updates are local.

Consider an exponential family,

$$q(\theta) = \exp[\lambda' T(\theta)] \quad \mu = E[T(\theta)]$$

where λ are the natural parameters, T the sufficient statistic, and μ the expectation parameters. Deep learning

$$\text{Minimize}_{\theta} \ell(\theta)$$

$$\theta \leftarrow \theta - \rho H_{\theta}^{-1} \nabla_{\theta} \ell(\theta)$$

and Bayesian learning are eerily similar.

$$\text{Minimize}_q E_{\theta \sim q} [\ell(\theta)] - H(q)$$

$$\lambda \leftarrow \lambda - \rho \nabla_{\mu} \left(E_{\theta \sim q} [\ell(\theta)] - H(q) \right)$$

For instance, Bayesian learning with $q(\theta) = N(m, 1)$ and a “global-to-local” approximation $E[\ell(\theta)] \approx \ell(\theta)$, gives gradient descent; $q(\theta) = N(m, S^{-1})$ gives Newton’s method.

**Interpretable comparison
of distributions and models**
A. Gretton et al. (NeurIPS 2019)

1. Integral probability metrics (IPM) compare measures p, q by looking at their difference

$$\text{IPM}(p, q) = \sup_{g \in H} \left| E_{x \sim p} g(x) - E_{y \sim q} g(y) \right|$$

while ϕ -divergences look at their ratio

$$D(p, q) = \int q(x) \phi \left(\frac{p(x)}{q(x)} \right) dx.$$

If H is the unit ball in a RKHS, the IPM is a maximum mean discrepancy (MMD)

$$\begin{aligned} f(x) &= \langle f, \phi(x) \rangle = \sum f_k \phi_k(x) \\ E_{x \sim p} [f(x)] &= \left\langle f, E_{x \sim p} [\phi(x)] \right\rangle = \langle f, \mu_p \rangle \\ \text{MMD}(p, q) &= \sup_{\|f\| \leq 1} E_{x \sim p} f(x) - E_{x \sim q} f(x) \\ &= \sup_{\|f\| \leq 1} \langle f, \mu_p - \mu_q \rangle \\ &= \|\mu_p - \mu_q\| \end{aligned}$$

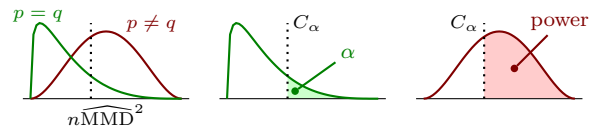
2. To test if two datasets (e.g., CIFAR-10 and CIFAR-10.1) come from the same distribution, compute their MMD.

$$\begin{aligned} \text{MMD}(p, q) &= \|\mu_p - \mu_q\|^2 \\ &= \langle \mu_p, \mu_p \rangle - 2\langle \mu_p, \mu_q \rangle + \langle \mu_q, \mu_q \rangle \\ &= E[\langle \phi X, \phi X' \rangle - 2\langle \phi X, \phi Y \rangle + \langle \phi Y, \phi Y' \rangle] \\ &= E[k(X, X') - 2k(X, Y) + k(Y, Y')]. \end{aligned}$$

When $p \neq q$, the distribution of $\widehat{\text{MMD}}(p, q)$ is asymptotically Gaussian, but when $p = q$, it depends on the kernel and the distribution of the data. Use *permutation testing* to select the test threshold; for the kernel, use a relevant representation $\varphi : \mathcal{X} \rightarrow \mathbf{R}^d$, e.g., a late hidden layer of a pretrained related classifier,

$$k(x, y) = k_{\text{top}}(\varphi(x), \varphi(y));$$

train the kernel to maximize the power $\text{MMD}^2(p, q) / \sigma_{H_1}(p, q)$.



A classifier to predict if a sample comes from p or q is a special case.

$$\begin{aligned} k(x, y) &= \frac{1}{4} \mathbf{1}_{f(x) > 0} \mathbf{1}_{f(y) > 0} \\ f : \mathcal{X} &\rightarrow \{\pm 1\} \text{ classifier} \\ \text{MMD}(p, q) &= |\text{accuracy} - \frac{1}{2}| \end{aligned}$$

3. The *witness function* (difference between kernel density estimates of the two distributions) defines the *un-normalized mean embedding*

$$\begin{aligned}\mu_p(v) &= E_{x \sim q}[k(x, v)] \\ \mu_q(v) &= E_{y \sim q}[k(y, v)] \\ \text{witness}(v) &= \mu_q(v) - \mu_p(v) \\ \text{UME}(p, q)^2 &= \frac{1}{J} \sum (\mu_q(v_j) - \mu_p(v_j))^2.\end{aligned}$$

Choose the v_j 's to maximize the (normalized) mean embedding.

4. The *Stein operator*

$$\begin{aligned}[T_p f](x) &= \frac{1}{p(x)} \frac{d}{dx} [f(x)p(x)] \\ E_p[T_p f] &= 0\end{aligned}$$

defines the *kernel Stein discrepancy*

$$\begin{aligned}\text{KSD}_p(q) &= \sup_{\|f\| \leq 1} \left| E_q[T_p f] - E_p[T_p f] \right| \\ &= \sup_{\|f\| \leq 1} E_q[T_p f],\end{aligned}$$

which can be computed as the norm of the *Stein witness function*

$$g(v) = E_{x \sim q} \left[\frac{1}{p(x)} \frac{d}{dx} [k(x, v)p(x)] \right]$$

which gives a goodness-of-fit test between a model p , known up to a normalization factor, and samples from a distribution q .

$$\begin{aligned}\text{KSD}_p^2(q) &= \|g\|^2 = E_{q \sim q} E_{y' \sim q} h_p(y, y') \\ h_p(x, y) &= s_p(x)' s_p(y) k(x, y) \\ &\quad + s_p(x)' \nabla_y k(x, y) + s_p(y)' \nabla_x k(x, y) \\ &\quad + \text{tr} \nabla_{xy} k(x, y) \\ s_p(x) &= \nabla_x \log p(x) \text{ score function}\end{aligned}$$

To select the test threshold, use the *wild bootstrap*

$$\frac{1}{n^2} \sum_{ij} w_i w_j h_p(y_i, y_j)$$

where $w_i = \pm 1$ is Bernoulli($\frac{1}{2}$).

5. The *finite set Stein discrepancy* test evaluates the Stein witness function at several points.

$$\text{FSSD}^2 = \frac{1}{d} \frac{1}{J} \sum \|g(v_j)\|^2$$

Geometric algebra E. Chisolm (2012)

Vector algebra, *i.e.*, vector manipulations in \mathbf{R}^3 , with scalar and cross products, looks unsatisfactory: it seems limited to dimension 3 (what about space-time?) and some operations look unnatural (the cross-product

needs a “handedness” convention and is not associative). Geometric algebra (Clifford algebras) can replace and generalize it.

A **geometric algebra** is a graded k -vector space $G = G_0 \oplus G_1 \oplus \dots$ with a (non-graded, non-commutative) k -algebra structure satisfying properties (i) to (iv) below. A **scalar** is an element of G_0 . A **vector** is an element of G_1 . An **r -blade** is a product of r anti-commuting vectors. An **r -vector** is a sum of r -blades. An **r -versor** is a product of r vectors. A **rotor** is a product of two invertible vectors.

- (i) $G_0 = k$.
- (ii) The square of every vector is a scalar.
- (iii) The symmetrized product of two vectors $(u, v) \mapsto \frac{1}{2}(uv + vu) = \frac{1}{2}((u+v)^2 - u^2 - v^2)$ is a non-degenerate bilinear form.
- (iv) $G_r = \text{span}\{r\text{-blades}\}$, *i.e.*, the r -vectors are the homogeneous elements of degree r .

Let $\langle \cdot \rangle_r : G \rightarrow G_r$ denote the projection. The product of $A_r \in G_r$ and $B_s \in G_s$ is a sum of terms of degrees $|r-s|, |r-s|+2, \dots, r+s$. One can define inner and outer products as the first and last terms:

$$\begin{aligned}A_r \lrcorner B_s &= \langle A_r B_s \rangle_{s-r} \\ A_r \llcorner B_s &= \langle A_r B_s \rangle_{r-s} \\ A_r \wedge B_s &= \langle A_r B_s \rangle_{r+s}\end{aligned}$$

Those operations have a simpler form for vectors, or vectors and versors.

$$\begin{aligned}a \lrcorner b &= b \llcorner a = \frac{1}{2}(ab + ba) \\ a_1 \wedge \dots \wedge a_r &= \frac{1}{r!} \sum_{\sigma} \text{sign}(\sigma) a_{\sigma(1)} \dots a_{\sigma(r)} \\ a A_r &= \langle a A_r \rangle_{r-1} + \langle a A_r \rangle_{r+1} = a \lrcorner A_r + a \wedge A_r \\ a \lrcorner A_r &= \frac{1}{2}(a A_r - (-1)^r A_r a) \\ a \wedge A_r &= \frac{1}{2}(a A_r + (-1)^r A_r a) \\ A_r \llcorner a &= (-1)^{r-1} a \lrcorner A_r \\ A_r \wedge a &= (-1)^r a \wedge A_r\end{aligned}$$

Those operations have geometric interpretations:

- $a_1 \wedge \dots \wedge a_r \neq 0$ iff a_1, \dots, a_r are linearly independent;
- $a \in \text{Span}\{a_1, \dots, a_r\}$ iff $a \wedge a_1 \wedge \dots \wedge a_r = 0$;
- $a \perp \text{Span}\{a_1, \dots, a_r\}$ iff $a \lrcorner (a_1 \wedge \dots \wedge a_r) = 0$;
- $\text{Span}\{a_1, \dots, a_r\} = \text{Span}\{b_1, \dots, b_r\}$ iff $\exists \lambda \ a_1 \wedge \dots \wedge a_r = \lambda b_1 \wedge \dots \wedge b_r$;
- $\text{Span}\{a_1, \dots, a_r\} \subset \text{Span}\{b_1, \dots, b_r\}$ iff $\exists c \ c \wedge a_1 \wedge \dots \wedge a_r = b_1 \wedge \dots \wedge b_r$.

Blades correspond to subspaces:

- $A \lrcorner B = 0$ iff $\exists a \in A \ a \neq 0, \ a \perp B$;
- $A \lrcorner B$ is the orthogonal complement of A in B ;
- $A^2 \in k$;
- $A^{-1} = A/A^2$;
- $A \subset B$ iff $AB = A \lrcorner B$;
- $A \perp B$ iff $AB = A \wedge B$.

One can define many more operations (I is the volume element).

Grade involution	$a^* = -a$ $A_r^* = (-1)^r A_r$
Reversion	$a^\dagger = a$ $(AB)^\dagger = BA$
Clifford conjugation	$A^\ddagger = A^*\dagger$
Scalar product	$A * B = \langle A^\dagger B \rangle$
Norm	$\ A\ ^2 = A * A$ (can be negative)
Dual	$A^\perp = A \lrcorner I^{-1} = AI^{-1}$
Commutator	$A \times B = \frac{1}{2}(AB - BA)$

Geometric algebra for computer science L. Dorst et al. (2009)

Geometric algebra represents geometric objects (points, lines, planes, circles spheres) and their transformations (translations, rotations, symmetries, Euclidean transformations, and even conformal transformations) as elements of a single set endowed with a dizzying array of products corresponding to geometric operation. There are three flavours, based on \mathbf{R}^3 , for linear geometry, on \mathbf{R}^4 , for affine geometry, on $\mathbf{R}^{4,1}$, for conformal geometry.

The book refuses to give clear, concise definitions of the operations and sets it uses and only provides verbose lists of special cases from which you are expected to get the general picture.

The **geometric product** is defined on $\bigwedge \mathbf{R}^n$ by

$$\begin{aligned} x^2 &= G(x, x) && \text{if } x \text{ is a vector} \\ \alpha\beta &&& \text{if } \alpha, \beta \text{ are scalars} \\ \alpha x &&& \text{if } \alpha \text{ is a scalar and } x \text{ a vector} \end{aligned}$$

and extended by distributivity, linearity and associativity (it is not commutative). For vectors, one can recover the scalar and external products as the symmetric and anti-symmetric parts of the geometric product,

$$xy = x \cdot y + x \wedge y.$$

Contrary to the outer product, the geometric product produces elements of mixed grade.

A **k -vector** is an element of $\bigwedge^k \mathbf{R}^n$. A **k -blade** is an outer product of k vectors (a k -vector is a linear combination of k -blades). A **versor** is a product of invertible vectors. A **rotor** is the product of an even number of unit vectors (*i.e.*, reflexions); in $\mathbf{R}^{n,0}$ and $\mathbf{R}^{n,1}$, it is the exponential of a bivector. Reflexions are $x \mapsto \hat{a}x\hat{a}^{-1}$; rotations are $x \mapsto RxR^{-1}$, where R is a rotor, e.g., $R = b/a$.

For *affine geometry*, let e_0, e_1, \dots, e_n be a basis of \mathbf{R}^{n+1} and use the embedding

$$\left\{ \begin{array}{ll} \mathbf{R}^n & \longrightarrow \mathbf{R}^{n+1} \\ \sum_{i \geq 1} a_i e_i & \longmapsto e_0 + \sum_{i \geq 1} a_i e_i. \end{array} \right.$$

For *conformal geometry*, let $e, e_1, \dots, e_n, \bar{e}$ be a basis of $\mathbf{R}^{n+1,1}$, set $o = \frac{1}{2}(e + \bar{e})$, $\infty = e - \bar{e}$ and use the embedding

$$\left\{ \begin{array}{ll} \mathbf{R}^n & \longrightarrow \mathbf{R}^{n+1,1} \\ p = \sum_{i \geq 1} a_i e_i & \longmapsto o + p + p^2 \infty. \end{array} \right.$$

Points in \mathbf{R}^n are represented by null vectors in $\mathbf{R}^{n+1,1}$ (*i.e.*, $p \cdot p = 0$; o and ∞ are also null vectors), and their product corresponds to the Euclidean distance

$$\frac{p}{-\infty \cdot p} \cdot \frac{q}{-\infty \cdot q} = -\frac{1}{2}(p - q)^2.$$

A vector $v \in \mathbf{R}^{n+1,1}$ can be interpreted in a dual way, $\{x : v \cdot x = 0\}$: null vectors are points; vectors with no o component are (dual) planes; in general, they are (dual) spheres.

Euclidean transformations are versors preserving ∞ . Lines are elements of the form point \wedge point $\wedge \infty$ or point \wedge direction $\wedge \infty$. Planes are of the form point \wedge point \wedge point $\wedge \infty$; replacing ∞ with a finite point gives circles and spheres.

A primer on reproducing kernel Hilbert spaces A.H. Manton and P.O. Amblard (2015)

A **finite-dimensional RKHS** is a subspace $V \subset \mathbf{R}^n$ endowed with an inner product. Its **kernel** is the unique matrix $k = (k_1, \dots, k_n) \in \mathbf{R}^{n \times n}$ satisfying the following three equivalent properties.

- (i) The k_i are in V and $\forall v \in V \quad \langle v, k_i \rangle = e'_i v$, *i.e.*, $\langle v, k_i \rangle$ is the i th coordinate of v ;
- (ii) $k = u_1 u'_1 + \dots + u_r u'_r$ where u_1, \dots, u_r is an orthonormal basis of V ;
- (iii) The k_i 's span V and $k_{ij} = \langle k_i, k_j \rangle$.

[Note that we do not need the scalar product of \mathbf{R}^n , only the coordinates.]

Graphically, a RKHS can be represented as the ellipsoid $\{v \in V : \langle v, v \rangle = 1\}$.

The columns of the kernel can be recovered as follows:

- Let $H_i = \{z \in \mathbf{R}^n : e'_i z = 1\} = \{z \in \mathbf{R}^n : z_i = 1\}$;
- If $V \cap H_i = \emptyset$, let $k_i = 0$;
- Otherwise, let $\tilde{k}_i = \text{Argmin}_{z \in V \cap H_i}$ and $k_i = \langle \tilde{k}_i, \tilde{k}_i \rangle^{-1} \tilde{k}_i$.

A **RKHS** is a subspace $V \subset \mathbf{R}^X$, with an inner product making it complete (a Hilbert space), such that the evaluation functionals

$$\text{ev}_x : \left\{ \begin{array}{ll} V & \longrightarrow \mathbf{R} \\ f & \longmapsto f(x), \end{array} \right.$$

for $x \in X$, be bounded (*i.e.*, continuous). Since $f \mapsto f(y)$ is continuous, from the Riesz representation theorem, there exists $k(\cdot, y) \in V$ such that $f(y) = \langle f, k(\cdot, y) \rangle$. The **kernel** $k : X \times X \rightarrow \mathbf{R}$ is positive semidefinite (often, when V is large enough, it is positive definite). Conversely, a positive semidefinite function $k : X \times X \rightarrow \mathbf{R}$ defines a RKHS $V = \text{Span}\{k(\cdot, y), y \in X\} \subset \mathbf{R}^X$.

Here are a few examples.

- The **Paley-Wiener space** is the space of band-limited functions $\mathbf{R} \rightarrow \mathbf{R}$,

$$f(t) = \frac{1}{2\pi} \int_{-a}^a F(\omega) e^{i\omega t} d\omega,$$

$F \in L^2$, for the scalar product $\langle f, g \rangle = \int_{-\infty}^{\infty} f g$; the kernel is

$$k(s, t) = \frac{\sin(a(s - t))}{\pi(s - t)}.$$

- The space of absolutely continuous functions $f : [0, 1] \rightarrow \mathbf{R}$, with $f(0) = 0$, whose derivative is square integrable, with the inner product $\langle f, g \rangle = \int_0^1 f' g'$; the kernel is $k(s, t) = \text{Min}(s, t)$;
- The **Bregman space** is the space of analytic and square-integrable functions on a domain $\Omega \subset \mathbf{C}$ with the inner product $\langle f, g \rangle = \int_{\Omega} f \bar{g}$; if Ω is the unit disk, the kernel is

$$k(z, w) = \frac{1}{\pi} \frac{1}{(1 - z\bar{w})^2}.$$

**The Ramanujan machine:
automatically generated conjectures
on fundamental constants
G. Raayoni et al. (2019)**

Given a fundamental constant (e.g., π , e , etc.), find 4 polynomials $\alpha, \beta, \gamma, \delta$ such that

$$\frac{\gamma(c)}{\delta(c)} = \text{GCF}(\alpha, \beta),$$

where the generalized continued fraction is

$$\text{GCF}(\alpha, \beta) = \alpha + \frac{\beta_1}{\alpha_1 + \frac{\beta_1}{\alpha_2 + \dots}}.$$

The meet-in-the-middle algorithm estimates the RHS at low precision, puts the result in a hash table, and check which LHS match; the precision is progressively increased.

Surprisingly, gradient descent to minimize

$$\left\| \frac{\gamma(c)}{\delta(c)} - \text{GCF}(\alpha, \beta) \right\|$$

can also lead to a solution:

- The minima are not isolated points but $(d - 1)$ -dimensional manifolds;
- All the minima seem to be global, and their error is zero;
- Start with a large number of point (500);
- Alternate gradient descent steps and “Coulomb repulsion” steps [?]
- To arrive at grid points, use gradient descent for the function $\sin^2(\pi x)$.

One can also allow α and β to be interlaced polynomials (e.g., different polynomials for even and odd terms) or transform the continued fraction with a simple function (e.g., $x \mapsto 1/x$).

Algorithms for reinforcement learning C. Szepesvári (2009)

2. Known MDP. A Markov decision process (MDP) (X, A, P_0) is the datum of a state space X , a set of actions A , and a mapping $P_0 : X \times A \rightarrow \mathcal{P}(X \times \mathbf{R})$ from state-action pairs to probability distributions on the next state and the reward. If $(X_{n+1}, R_{n+1}) \sim P_0(\cdot | X_n, A_n)$, the discounted reward is $R = \sum_{t \geq 0} \gamma^t R_{t+1}$.

An *episodic* MDP has a terminal (absorbing) state.

A *bandit* is an 1-state MDP.

A deterministic (resp. stochastic) *policy* is the choice of an action for each state, $A_t = \pi(X_t)$, resp. $A_t \sim \pi(\cdot | X_t)$.

A **Markov reward process** (MRP) is an MDP without actions – or, equivalently, with a stationary policy.

The value function of a MRP

$$V^\pi(x) = E\left[\sum \gamma^t R_{t+1} \mid X_0 = x\right]$$

$$Q^\pi(x, a) = E\left[\sum \gamma^t R_{t+1} \mid X_0 = x, A_0 = a\right]$$

satisfies the Bellman equation

$$\begin{aligned} V^\pi(x) &= r(x, \pi(x)) + \gamma \sum_{y \in X} P(x, \pi(x), y) V^\pi(y) \\ &= (T^\pi V^\pi)(x). \end{aligned}$$

Likewise, the optimal value function V^* (or Q^*) of an MDP satisfies

$$\begin{aligned} V^*(x) &= \sup_{a \in A} r(x, a) + \gamma \sum_{y \in X} P(x, a, y) V^*(y) \\ &= (T^* V^*)(x). \end{aligned}$$

It can be obtained by **value iteration**, $V_{k+1} = T^* V_k$, or $Q_{k+1} = Q^* V_k$.

Policy iteration uses the state-action value function Q instead of the state value function V and iterates the following two steps:

- Compute the value Q^{π_k} of the policy π_k ;
- Compute the greedy policy π_{k+1} for this value function Q^{π_k} .

3. Unknown MDP. The value of a state can be of independent interest: probability of reaching a state, or expected time until we reach a state.

TD(0) moves the estimation of the value of the current state X_t towards a target.

$$\text{target} = R_{t+1} + \gamma V_t(X_{t+1})$$

$$\delta_{t+1} = \text{target} + \gamma V_t(X_{t+1})$$

$$V_{t+1}(x) = \begin{cases} V_t(X_t) + \alpha \delta_{t+1} & \text{if } x = X_t \\ \text{unchanged} & \text{otherwise} \end{cases}$$

The step size α_t should satisfy the Robbins-Monroe condition, e.g., $\alpha_t \propto t^{-\eta}$, $\frac{1}{2} < \eta \leq 1$, with $\eta = 1$ for good asymptotic behaviour, and η close to $\frac{1}{2}$ for good small sample properties – and $\eta = 0$ in practice.

TD(0) can be used for off-policy learning.

Every-visit-Monte-Carlo is similar, but the target is the discounted reward until the end of the episode. TD(0) converges faster, but struggles with delayed rewards.

It is actually not necessary to wait until the end of the episode: one can keep track of the eligibility of each state for the next rewards; the eligibility is increased by 1 each time the state is reached and then exponentially decays. TD(λ) lets it decay faster than the discounting faster; TD(1) is every-visit MC.

$$\begin{aligned}\delta_{t+1} &= R_{t+1} + \gamma V_t(X_{t+1}) - V_t(X_t) \\ z_{t+1}(x) &= \mathbf{1}_{x=X_t} + \gamma \lambda z_t(x) \\ V_{t+1}(x) &= V_t(x) + \alpha \delta_{t+1} z_{t+1}(x) \\ z_0(x) &= 0\end{aligned}$$

Variants of the eligibility traces include $z_{t+1}(x) = \text{Max}(\mathbf{1}_{x=X_t}, \gamma \lambda z_t(x))$, corresponding to *first-visit Monte Carlo* – it may perform better. The optimal value of λ is determined by trial and error.

For large state spaces, the value function can be approximated from state features, $\phi(x)$, e.g., $V_\theta(x) = \theta' \phi(x)$, with

- $\phi_i(x) = \exp(-\eta \|x - x^{(i)}\|^2)$ for manually-chosen points $x^{(i)}$ (if these points are model parameters, this is an **RBF network**);
- ϕ_i 's rescaled so that $\sum \phi_i \equiv 1$ (*averager*);
- Kernel smoothing, spline smoothing, Gaussian process (GP) regression, regression tree.

With function approximation, the eligibility traces are for the coordinates of θ instead of the states, and they use the gradient $\nabla_\theta V_\theta$ instead of the indicator function $\mathbf{1}_{X_t=x}$.

$$\begin{aligned}\delta_{t+1} &= R_{t+1} + \gamma V_{\theta_t}(X_{t+1}) - V_{\theta_t} \\ z_{t+1} &= \nabla_\theta V_\theta(X_t)|_{\theta=\theta_t} + \gamma \lambda z_t \\ \theta_{t+1} &= \theta_t + \alpha_t \delta_{t+1} z_{t+1}\end{aligned}$$

There is no convergence guarantee for TD(λ) with function approximation if the model is not linear or for off-policy learning.

Gradient TD learning directly minimizes the loss function

$$J(\theta) = \|V_\theta - \Pi T V_\theta\|_2^2$$

where T is the Bellman operator and Π the best approximation of its argument in the set of functions considered. The gradient of J is complicated: write it using θ and some other quantity $w(\theta)$; successively update θ (assuming w fixed) and w . There are several ways of choosing such a decomposition (GTD2, TDC, LS, etc.).

4. There are several types of reinforcement learning problems:

- No interaction: the data has already been collected;

- *Active*: we can explore freely – the system is running, but not live yet;
- *Online*: we already want good rewards – the system is already live – we can explore, but not as freely as we want.

For online learning of bandits, ε -greedy (with decreasing ε) or *Boltzman exploration* $\pi(a) \propto \exp \beta Q(a)$ are popular.

UCB₁ takes the action with the best upper confidence bound.

$$\begin{aligned}U(a) &= r(a) + R \sqrt{\frac{2 \log t}{n(a)}} \\ r(a) &= \text{average reward for action } a \\ n(a) &= \text{number of times action } a \text{ was chosen} \\ R &= \text{variance or range of the rewards for } a\end{aligned}$$

For Bernoulli rewards, we can explicitly compute the policy maximizing the expected reward (Gittins index).

$$\begin{aligned}R(a) &\sim \text{Bernoulli}(p_a) \\ p_a &\sim \text{Beta}(\alpha_a, \beta_a) \\ \alpha_a, \beta_a &\sim \text{some prior}\end{aligned}$$

For active learning of bandits, compute upper and lower bounds for each action and discard an action a if

$$U_t(a) < \text{Max}_{\substack{a' \in \mathcal{A} \\ a' \neq a}} L_t(a').$$

While it is possible to recover a deterministic MDP (find the closest state with an unexplored action), there is no known (reasonable) algorithm to recover a stochastic MDP.

The UCRL2 algorithm, for online learning of MDPs, constructs confidence intervals for the transition probabilities and the reward function, defining a set C_t of plausible MDPs, and updates the policy when those estimates are precise enough, to maximize the discounted reward.

$$\pi^*, M^* = \underset{\substack{\pi \text{ policy} \\ M \in C_t}}{\text{Argmax}} \rho^\pi(M)$$

4.3 Q-learning is TD(0) for Q instead of V , for off-policy learning (TD(λ) is on-policy), for ε -greedy or Boltzman exploration. The MDP can sometimes be simplified by separating the deterministic effects of an action from its stochastic effect (“after-effect”).

Q-learning with function approximation is not guaranteed to converge for non-linear approximation or off-policy learning.

4.4 Actor-critic methods generalize policy iteration (the actor is the policy, the critic the value function): they update the policy before it has been completely evaluated; they may oscillate: they keep track of the best performing policy. **SARSA** is an extension of

TD(0) to Q instead of V .

$$\text{target} = R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q(y_{t+1}, a') \quad \text{Q-learning}$$

$$\text{target} = R_{t+1} + \gamma Q(y_{t+1}, A'_{t+1}) \quad \text{SARSA}$$

The other value estimation algorithms for V can be extended to Q : TD(λ), LSTD(λ), etc.

The actor is updated less often, either as a greedy policy (which can be computed on the fly) or, for parametrized stochastic policies, e.g., Boltzman

$$\pi_\omega(a|x) \propto \exp w' \xi(x, a),$$

where ξ are features, or Gaussian,

$$\pi_\omega(a, x) \sim N(g_\omega(x, a), \beta_\omega(x, a)^2 I),$$

with gradient ascent

$$\nabla_\omega \rho = E[G(\omega)]$$

$$G(\omega) = (Q^{\pi_\omega}(X, A) - h(X))\psi_\omega(X, A)$$

$$\psi_\omega = \nabla_\omega \log \pi_\omega \quad \text{score function}$$

$$h(X) = V(X) \quad \text{or an arbitrary function,} \\ \text{for variance reduction}$$

REINFORCE updates the parameters at the end of the episodes.

The score function can be used as features,

$$Q_\theta(x, a) = \theta' \psi_\omega(x, a).$$

The natural actor-critic uses the natural gradient but has simpler updates.

Distral:
robust multitask reinforcement learning
Y.W. Teh et al.

For multitask learning, learn the tasks separately, with a penalty to make them closer to an average task, computed by distillation.

Large-scale optimal transport
and mapping estimation
V. Seguy et al.

The dual of the regularized Kantorovich relaxation

$$\underset{\substack{\text{pr}_{1*} \pi = \mu \\ \text{pr}_{2*} \pi = \nu}}{\text{Minimize}} \quad E_{(X,Y) \sim \pi} c(X, Y) + \varepsilon R(\pi)$$

$$\text{with } R(\pi) = \int \left(\log \frac{p\pi}{d(\mu \times \nu)} - 1 \right) d\pi$$

$$\text{or } R(\pi) = \int \left(\frac{p\pi}{d(\mu \times \nu)} \right)^2 d\mu d\nu$$

is an unconstrained optimization problem

$$\underset{u, v}{\text{Maximize}} \quad E_{\substack{X \sim \mu \\ Y \sim \nu}} u(X) + v(Y) - F_\varepsilon(u(X), v(Y))$$

which can be solved by stochastic gradient descent, if u and v are modeled by neural nets (the Sinkhorn algorithm is quadratic and only works with discrete distributions); the coupling π can then be turned into a map f (modeled with a neural net and estimated by SGD) by *barycentric projection*

$$f(x) = \underset{y \in \mathcal{Y}}{\text{Argmin}} \quad E_{Y \sim \pi(X, \cdot)} [d(z, Y)].$$

Machine learning applications include *domain adaptation* (transfer learning: aligning different datasets), shape matching, colour transfer, data assimilation (aligning biased model(s) and data).

Sliced Gromov-Wasserstein
T. Vayer et al.

The **sliced Wasserstein distance** is the average of the Wasserstein distance of projections on random 1-dimensional subspaces.

The **Gromov-Wasserstein distance** between $\mu = \sum a_i \delta_{x_i} \in \mathcal{P}(\mathbf{R}^p)$ and $\nu = \sum b_j \delta_{y_j} \in \mathcal{P}(\mathbf{R}^q)$ for the dissimilarity measures

$$c_X : \mathbf{R}^p \times \mathbf{R}^p \longrightarrow \mathbf{R}_+$$

$$c_Y : \mathbf{R}^q \times \mathbf{R}^q \longrightarrow \mathbf{R}_+$$

is

$$\text{GW}_2 = \underset{\pi \in \Pi(a, b)}{\text{Min}} \sum_{ijkl} |c_X(x_i, x_k) - c_Y(y_j, y_\ell)|^2 \pi_{ij} \pi_{kl}.$$

The 1-dimensional uniform case, with squared Euclidean distances, is easy: sort the x_i 's and y_j 's (in increasing or decreasing order, whichever is better).

Pushing the right boundaries matters!
Wasserstein adversarial training
for label noise
B.B. Damodaran et al.

For an observation (y, x) , **virtual adversarial training** adds a regularization term $D(p_\theta(x) \| p_\theta(x + r))$ where p_θ are the predicted class probabilities (when computing the gradient wrt θ , fix the first argument and only differentiate the second) and r , with $\|r\| \leq \varepsilon$, is the direction in which the divergence increases the most (the dominant eigenvector of the Hessian, $D(p(x) \| p(x + r)) \sim \frac{1}{2} r' H r$, obtained by power iteration – one step is enough).

Replace the KL divergence with the Wasserstein distance (with entropic regularization), with a cost reflecting similarities between classes (e.g., from word2vec).

The regularization is then more important for classes that should not be confused, that are less likely to be affected by noise.

Other ways of dealing with noisy labels include robust methods, data cleaning (with another network), adding a noise layer on top of the softmax, and learning the *noise transition matrix*.

The Wasserstein transform
F. Mémoli et al (2019)

A point cloud $X \subset \mathbf{R}^n$ can be seen as a uniform distribution α on the metric space (X, d) . The local ε -truncation of α is the probability measure

$$m_\alpha^\varepsilon(x) \propto \alpha|_{B_\varepsilon(x)}.$$

The **Wasserstein transform** is a new distance on X

$$d_\alpha^\varepsilon(x, y) = d_{W,1}(m_\alpha^\varepsilon(x), m_\alpha^\varepsilon(y))$$

where $d_{W,1}$ is the ℓ^1 Wasserstein distance. The iterated Wasserstein transform generalizes the mean-shift algorithm and has a denoising effect.

An introduction to topological data analysis: fundamental and practical aspects for data scientists
F. Chazal and B. Michel (2017)

To turn persistence diagrams into features, use persistence landscapes, or build a persistence diagram *kernel*.

Information-geometric optimization algorithms: a unifying picture via invariance principles
Y. Ollivier et al. (2017)

Gradient descent on $F(\theta) = \mathbb{E}_{X \sim p_\theta} [f(\theta)]$ (where the p_θ are, e.g., Gaussians) converges to a Dirac mass at the minimum of f :

- Rewrite f using quantiles, to make the algorithm invariant to monotonic transformations;
- Use the natural gradient;
- Approximate the expectation using quantiles (as CMA-ES or CEM).

Riemann manifold Langevin and Hamiltonian Monte Carlo
M. Girolami et al.

The Metropolis-adjusted Langevin algorithm is a Metropolis-Hastings Markov chain with a drift term (the Langevin diffusion is unbiased when the step size tends to zero – for finite step sizes, a MH accept-reject step is needed – HMC has the same problem).

One-step HMC updates are Langevin updates.

On a Riemannian manifold, one can leverage the metric:

$$\begin{aligned} d\theta &= \frac{1}{2} \nabla_\theta \ell(\theta) dt + db && \text{flat} \\ d\theta &= \frac{1}{2} \tilde{\nabla} \ell dt + \tilde{db} && \text{Riemann} \\ \tilde{\nabla} \ell &= G^{-1} \nabla \ell && \text{Natural gradient} \\ \tilde{db} &= \dots dt + G^{-1/2} db. \end{aligned}$$

In the Riemannian Brownian motion, the second term accounts for the curvature, and the first for the change in curvature.

The Hamiltonian is simpler,

$$\begin{aligned} H(\theta, p) &= -\ell(\theta) + \frac{1}{2} \log(2\pi)^D |G| + \frac{1}{2} p' G^{-1} p \\ \frac{d\theta_i}{d\tau} &= \frac{\partial H}{\partial p_i} \\ \frac{dp_i}{d\tau} &= -\frac{\partial H}{\partial \theta_i} \end{aligned}$$

but the naive discrete integrator is not volume-preserving: use an implicit integrator instead.

Topological denoising: strengthening the topological signal
J. Kloeke and G. Carlsson (2018)

A few steps of the **mean-shift** algorithm have a denoising effect on a point cloud X . If there is more noise, take a subset $S_0 \subset X$ (e.g., 10%) and progressively move them in the direction of the gradient of

$$F_n(x) = f_D(x) + \omega f_{S_n}(x)$$

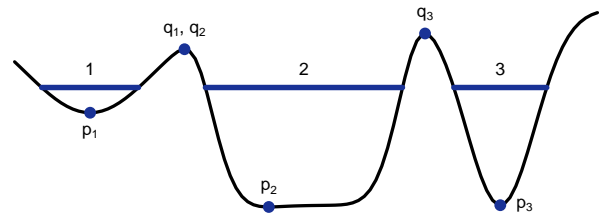
where f_D and f_{S_n} are density estimators (Gaussian, with the same bandwidth σ) of D and S_n and $\omega \in [.1, .5]$. The first term is from the mean-shift algorithm; the second keeps the points of S_n away from each other.

A topological regularizer for classifiers via persistent homology
C. Chen et al. (2019)

The **topological complexity** of the decision boundary $f^{-1}(0)$ of a classifier $f : X \rightarrow \mathbf{R}$ is $\sum_c \rho(c)^2$ where the **robustness** $\rho(c)$ of connected component c of $f^{-1}(0)$ is

$$\text{Min}\{|f(p_c)|, |f(q_c)|\},$$

where p_c and q_c are the critical points corresponding to the birth and death of the component (0th dimensional persistent homology – the *persistence* is $f(q_c) - f(p_c)$). It can be used as a regularizer (TDA usually yields features); to compute its gradient, use a piecewise approximation of the classifier function.



Connectivity-optimized representation learning via persistent homology
C.D. Hofer et al. (2019)

Use connectivity-optimized representation learning via persistent homology.

On characterizing the capacity of neural networks using algebraic topology
W.H. Gus and R. Salakhutdinov

The minimum number of neurons depends on the “geometric complexity” of the dataset. Conversely, the “topological capacity” of a neural net can be estimated with topological data analysis (TDA).

Neural persistence: a complexity measure for deep neural networks using algebraic topology
B. Rieck et al. (2019)

The *neural persistence* of a layer is the L^p norm of the 0th homology persistence diagram of the corresponding bipartite graph for the filtration induced by the weights. Regularized networks (batchnorm, dropout) have a higher (normalized) persistence.

Weisfeiler-Lehman graph kernels
N. Shervashidze et al. (2011)

The **Weisfeiler-Lehman test** for graph isomorphism between G and G'

- Augments the node labels with the (sorted) labels of the neighbours;
- Compresses the label multisets into shorter labels (e.g., with a hash function, or with a bijection with $\llbracket 1, k \rrbracket$);
- Repeats until the label sets of G and G' differ or a timeout is reached.

Those new labeled graphs $G_0, G_1, G_2, \dots, G'_0, G'_1, \dots$ can be used to define graph kernels as

$$\sum_{i=0}^h k(G_i, G'_i) \quad \text{or} \quad k(G_h, G'_h),$$

where k is an easy-to-compute kernel, e.g., the number of matching node labels, or the number of edge pairs whose extremities have matching labels.

A persistent Weisfeiler-Lehman procedure for graph classification
B. Rieck et al. (2019)

Label propagation (Weisfeiler test) defines a distance between adjacent vertices

$$d^h(u, v) = \mathbf{1}_{\ell_u^{h-1} \neq \ell_v^{h-1}} + d(\ell_u^h, \ell_v^h) + 1$$

where ℓ_u^h is the label multiset of node u after h steps and the distance between multisets is the L^p distance between their count vectors. The distance d^h defines a filtration on the graph, for which we can compute the persistence diagram, which can be added (at each step) to the WL features (to define a graph kernel).

A tutorial on variational Bayesian inference
C. Fox and S. Roberts

We want to approximate a probability distribution $P(x)$, e.g., from a graphical model, $P(x) = \prod_I \Phi(x_I)$,

with a **mean-field approximation**, i.e., $Q(x, \theta) = \prod_i Q_i(x_i, \theta)$, by minimizing $\text{KL}(Q\|P)$.

$$\begin{aligned} \text{KL}(Q\|P) &= \int Q \log \frac{Q}{P} \\ &= \int Q(x) \log \frac{Q(x)}{P(x|\text{data})} dx \\ &= \int Q(x) \log \frac{Q(x)P(\text{data})}{P(x, \text{data})} dx \\ &= \int Q(x) \log Q(x) dx - \int Q(x) \log P(x, \text{data}) dx + \\ &\quad \int Q(x) dx P(\text{data}) \\ &= -H(Q) - \int E(x, \text{data}) Q(x) dx + P(\text{data}) \end{aligned}$$

$$H(Q) = - \int Q(x) \log Q(x) dx \quad \text{Shannon entropy}$$

$$E(x, \text{data}) = \log P(x, \text{data}) \quad \text{Energy}$$

$$L = \langle E \rangle + H \quad \text{Lower bound on } P(\text{data})$$

Since $Q(x) = \prod_i Q_i(x_i)$, we have $H(Q) = \sum_i H(Q_i)$ and

$$\begin{aligned} \langle E \rangle &= \int Q(x_i) \int Q(\bar{x}_i) E(x, \text{data}) d\bar{x}_i dx_i \\ &= \int Q(x_i) \log Z Q^*(x_i) dx_i \\ &= \int Q(x_i) \log Q_i^*(x_i) dx_i + \log Z \end{aligned}$$

where

$$Q_i^* = \frac{1}{Z} \exp \int Q(\bar{x}_i) E(x, \text{data}) d\bar{x}_i.$$

Therefore,

$$\begin{aligned} L &= \langle E \rangle + H \\ &= \int Q(x_i) \log Q_i^*(x_i) dx_i + \log Z - \sum_j \int Q(x_j) \log Q(x_j) dx_j \\ &= \int Q(x_i) \log \frac{Q_i^*(x_i)}{Q_i(x_i)} + \log Z - \sum_{j \neq i} H(Q_j) \\ &= -\text{KL}(Q_i\|Q_i^*) + \log Z - \sum_{j \neq i} H(Q_j). \end{aligned}$$

The lower bound L is maximal when $\text{KL}(Q_i\|Q_i^*) = 0$, i.e., when $Q_i = Q_i^*$. This gives an EM-like algorithm

$$Q_i(x_i) \leftarrow \frac{1}{Z} \exp \langle \log P(x_i, \bar{x}_i | \text{data}) \rangle_{Q(\bar{x}_i)}$$

where $\bar{x}_i = x_{I \setminus \{i\}}$ or $\bar{x}_i = \text{MarkovBlanket}(x_i)$.

Automatic differentiation variational inference A. Kucukelbir et al.

Variational inference (VI) requires model-specific derivations and implementations, but the gradient of the VI objective can be written as an expectation (involving $\nabla_{\theta} p_{\theta}(x)$, which can be computed by automatic differentiation (AD)) over the variational distribution q_{ϕ} and approximated with Monte Carlo methods (ADVI). ADVI is not limited to mean-field approximations: full-rank approximations do not underestimate the posterior variance as much.

Examples include nonnegative matrix factorization, e.g., Gamma-Poisson

$$\begin{aligned}\theta_{uk} &\sim \text{Gamma}(a, b) \\ \beta_{ik} &\sim \text{Gamma}(c, d) \\ y_{ui} &\sim \text{Poisson}(\theta'_u \beta_i)\end{aligned}$$

or Dirichlet-exponential-Poisson

$$\begin{aligned}\theta_u &\sim \text{Dir}(\alpha) \\ \beta_{ik} &\sim \text{Exp}(\lambda) \\ y_{ui} &\sim \text{Poisson}(\theta'_u \beta_i)\end{aligned}$$

and models with a sparsifying *automatic relevance determination* (ARD) prior, e.g., a linear model

$$\begin{aligned}\sigma &\sim \text{InvGamma}(a, b) \\ \alpha_i &\sim \Gamma(c, d) \\ w_i &\sim N(0, \sigma^2 / \alpha_i) \\ \varepsilon &\sim N(0, \sigma^2) \\ y &= w'x + \varepsilon\end{aligned}$$

or probabilistic PCA (PPCA), or supervised PPCA.

Deep set prediction networks Y. Zhang et al.

To provide sets as input to a neural network, use an invariant function $\{x_i\}_i \mapsto \sum_i g(x_i)$, where g is a neural net.

To compare sets, use an optimal alignment

$$\ell(\hat{y}, y) = \min_{\sigma \in \mathfrak{S}_n} \|\hat{y} - \sigma \cdot y\|^2.$$

To output sets, decode the latent representation into the set whose encoding is closest to the desired latent representation (this requires an optimization). For variable-sized sets, pad the sets to a fixed size and add a mask feature.

This can also be used for an autoencoder.

$$\begin{aligned}\{x_i\}_i &\mapsto z = \sum_i g(x_i) \mapsto \text{Argmin}_{\{y_i\}} \|x - \sum g(y_i)\|^2 \\ x = \sum_i g(y_i) &\longleftarrow \{y_i\}_i\end{aligned}$$

Provably robust deep learning via adversarially trained smoothed classifiers H. Salman

Randomized smoothing transforms a classifier to retain the most likely class in a neighbourhood of the query

$$g(x) = \text{Argmax}_c \mathbb{P}_{\varepsilon \sim N(0, \sigma^2 I)} [f(x + \varepsilon) = c];$$

it makes the classifier robust to random (non-adversarial) perturbations.

It can be generalized to *soft classifiers*, which output a probability distribution on classes rather than a single class.

Adversarial training uses the gradient at adversarial perturbations instead of the actual observations; It can be applied to smoothed (soft) classifiers.

Kervolutional neural networks C. Wang et al.

The convolution $f(x) = x * w = (\langle x_{(i)}, w \rangle)_i$, where $x_{(i)}$ is the circular shift of x by i elements, can be kernalized

$$g(x) = x \circledast w = (\langle \phi(x_{(i)}), \phi(w) \rangle)_i = (\kappa(x_{(i)}, w))_i$$

where κ is a fixed kernel (polynomial or RBF).

DeepRED: deep image prior powerd by RED G. Mataev et al. (2019)

The **deep image prior** (DIP) addresses inverse imaging problems (denoising, deblurring, inpainting, super-resolution, tomographic reconstruction) with a neural net T_{θ} :

$$\begin{aligned}\text{Find} & \quad x \text{ reconstructed image} \\ & \quad \theta \text{ model parameters} \\ \text{To minimize} & \quad \|Hx - y\|_2^2 \\ \text{Such that} & \quad x = T_{\theta}(z)\end{aligned}$$

where H is the (known) corruption (the identity for denoising, a convolution for deblurring, etc.), y is the corrupted image and z a fixed random vector.

While this already provides an implicit regularization, one can add an explicit regularization and minimize

$$\|Hx - y\|_2^2 + \lambda x'(x - f(x))$$

where f is a fixed denoiser (for a reasonable choice of f , the gradient of the regularizer has a simple form, $x - f(x)$).

Visualizing and measuring the geometry of BERT A. Coenen et al.

BERT provides contextual word embeddings whose (squared) distance matches the distance on the parse

tree; the attention matrices contain similar information. Any tree has a power-2 embedding (*i.e.*, $\|f(x) - f(y)\|^2 = d(x, y)$) in \mathbf{R}^{n-1} :

$$\begin{aligned} f(t_0) &= 0 \\ f(t_i) &= e_i + f(\text{parent}(t_i)) \end{aligned}$$

where the e_i 's form an orthonormal basis (or are random unit vectors in a high-dimensional space). The semantic information of the embeddings and the polysemy of the words can be visualized by selecting a word, picking 1000 Wikipedia sentences containing it, computing the embeddings, and plotting them with UMAP.

***MixMatch: a holistic approach
to semi-supervised learning***
D. Berthelot et al.

To leverage unlabeled data, semi-supervised learning can use

- **Consistency regularization**, to ensure the output of the classifier remains the same after data augmentation (image patches, orientation, deformations, noise, etc.),
- **Entropy minimization**, to prevent the decision boundary from going through high-density regions, by minimizing the entropy $p_{\text{model}}(y|x)$ for unlabeled data,

in addition to the traditional (supervised) regularizations:

- L^2 regularization, often implemented as a **weight decay**, to prevent the model from memorizing the training data;
- **MixUp**, which trains the model on convex combinations of inputs and labels.

MixMatch combines those ideas:

- Data augmentation on both labeled and unlabeled data;
- Label distribution guess for unlabeled observations by averaging the predicted distributions after data augmentation;
- Distribution sharpening, $q_i \propto p_i^{1/T}$, $T < 1$;
- Train the model on convex combinations of labeled and/or unlabeled observations, but with a different loss function depending on whether the labeled or unlabeled observation is closer.

***Phase transitions of spectral initialization
for high-dimensional nonconvex estimation***
Y.M. Lu and G. Li

Phase retrieval is the problem of estimating ξ given noisy measurements $y_i = (a'_i \xi)^2 + \varepsilon_i$; more generally, one may want to estimate ξ from measurements $y_i \sim f(\cdot | a'_i \xi)$.

Spectral initialization suggests to initialize ξ to the first eigenvector of the variance matrix of the a_i 's [*i.e.*, where the information is the “densest”: that is not

where the solution is, but that is where you will see the most clearly where to go – provided there is enough data].

Stochastic bouncy particle sampler
A. Pakman et al.

The **Bouncy particle sampler** (BPS) samples from $p(w) \propto e^{-U(w)}$, $w \in \mathbf{R}^f$, by adding a random velocity vector $v \in \mathbf{S}^{d-1}$

- w follows the direction v (straight lines);
- v is relected along ∇U^\perp , following an inhomogeneous Poisson process of intensity $\lambda = (v \cdot \nabla U)_+$:

$$v \leftarrow v - 2 \frac{v \cdot \nabla U}{\|\nabla U\|^2} \nabla U;$$

- Occasionally resample v to ensure ergodicity.

With noisy gradients, the Poisson intensity becomes stochastic (doubly stochastic process, aka Cox process) – the bounces are more frequent and mixing is slower. To sample from it, find an upper bound on λ , e.g., using a local regression $\lambda \sim t$.

(**Zig-zag Monte Carlo** is similar, with $v \in \{\pm 1\}^d$, and only changes one coordinate at a time.)

***The bouncy particle sampler:
a nonreversible rejection-free
Markov chain Monte Carlo method***
A. Bouchard-Côté et al. (2018)

***Adaptive hard thresholding
for near optimal consistent robust regression***
A.S. Suggala et al. (2019)

Robust linear regression by discarding observations with “large” residual:

- Iteratively estimate the regression coefficients on the set of uncorrupted observations;
- Use an adaptive threshold to detect outliers;
- Add some noise when deciding whether to include an observations.

The (slower) Huber loss also works well.

The cult of statistical significance
S.T. Ziliak and D.N. McCloskey (2009)

The p -value is easy to compute, but it is not the *loss function* you want – prefer something involving the effect size. For instance, $X_1 \sim N(\mu = 5, \sigma = 0.5)$ has a better p -value than $X_2 \sim N(\mu = 20, \sigma = 10)$, 10^{-23} versus 10^{-1} (for $H_0 : X > 0$), but $P(X_1 > X_2) = 0.07\dots$

***Uncertainty propagation
with functionally correlated quantities***
M. Giordano (2016)

The **Mesurements.jl uncertainty propagation** package handles *functional correlations*, *i.e.*, deterministic relations $y = f(x)$; for instance, given $x = x_0 \pm \sigma$, it estimates $x - x = 0 \pm 0$, not $x - x = 0 \pm 2\sigma$.

**On cross-validation
for sparse reduced-rank regression
Y. She and H. Tran (2018)**

Consider a jointly sparse and low-rank linear model

$$\begin{matrix} m \\ \boxed{Y} \end{matrix} = \begin{matrix} p \\ \boxed{X} \end{matrix} \times \begin{matrix} m \\ \boxed{B} \end{matrix} + \varepsilon$$

where B is both low-rank and row-sparse (some of the rows are zero).

To choose the regularization parameters, e.g., λ for the lasso, one can use cross-validation

- On λ , the scale of the penalty $\lambda \|\beta\|_1$, but a different number of variable may be selected in different folds;
- Or on c , the corresponding constraint $\|\beta\|_1 \leq c$, but this is difficult to solve.

Instead, use cross-validation on both

- Which variables are retained by the model;
- The (low-rank) subspace spanned by the coefficient matrix

(each can be encoded as a matrix, S and U , which can be recovered from their product SU).

[I am not sure how to use that in practice: with λ , we can just try a few values, on a grid, but here, we have a combinatorial choice ($p!$ subsets of predictors) and a subspace.]

The optimal rank and sparsity can be estimated using

- Some information criterion (AIC, BIC, etc. – but it is not clear which one to use);
- Or cross-validation.

**Natural analysts in adaptive data analysis
T. Zrnic and M. Hardt**

Data analysis is no longer limited to 1- or 2-step procedures (statistical tests, post-selection inference): by progressively interrogating the data, the analyst's knowledge evolves – it is a discrete dynamical system. *Differential privacy* can help limit overfitting.

**Minimax optimal rates
for Mondrian trees and forests
J. Mourtada et al. (2018)**

A *Mondrian process* is a distribution on infinite nested trees (partitions of $[0, 1]^d$) defined from iid sequences of random variables $e \sim \text{Exp}(1)$ used to select the dimension to split, by comparing it with the side lengths of the box to split

$$\tau_i = \frac{e_i}{\text{length}_i} \quad i = \underset{i}{\text{Argmin}} \tau_i \quad \tau = \underset{i}{\text{Min}} \tau_i,$$

and $u \sim U(0, 1)$ used to choose the threshold to split. A *Mondrian forest* is a *purely random forest* whose trees are sampled from a Mondrian process.

**Implicit generation and modeling
with energy-based models
Y. Du and I. Mordatch**

Energy-based models, $p(x) \propto e^{-E(x)}$, E unknown, can be estimated by MCMC

$$\begin{aligned} \ell(\theta) &= \underset{x \sim \text{data}}{\text{E}} [-\log p_\theta(x)] \\ &= \underset{x \sim \text{data}}{\text{E}} [E_\theta(x) - \log Z(\theta)] \\ \nabla \ell(\theta) &= \underset{x^+ \sim \text{data}}{\text{E}} [\nabla_\theta E_\theta(x^+)] - \underset{x^- \sim p_\theta}{\text{E}} [\nabla_\theta E_\theta(x^-)] \end{aligned}$$

**Meta-learning how to forecast time series
T.S. Talagala et al. (2018)**

Train a random forest to forecast the best time series forecasting algorithm (SARIMA, exponential smoothing, random walk) using time series features (length, trend, seasonality, linearity, curvature, spikiness, autocorrelation, stability, lumpiness, spectral entropy, Hurst exponent, nonlinearity test, ETS model parameters, unit root tests). Implementation: **seer**.

**On fast convergence of proximal algorithms
for SQRT-lasso optimization:
don't worry about its nonsmooth loss function
X. Li et al.**

The optimal regularization parameter for the lasso regression depends on the noise variance

$$\underset{\theta}{\text{Argmin}} \frac{1}{n} \|y - X\theta\|_2^2 + \lambda \|\theta\|_1 \quad \lambda \approx \sigma \sqrt{\frac{\log d}{n}}$$

while that for the *SQRT-lasso* (whose loss function is the *unsquared* ℓ^2 norm) does not

$$\underset{\theta}{\text{Argmin}} \frac{1}{\sqrt{n}} \|y - X\theta\|_2 + \lambda \|\theta\|_1 \quad \lambda \approx \sqrt{\frac{\log d}{n}}.$$

The **proximal Newton** algorithm solves this problem by using a second order expansion of the loss function (leaving the penalty intact)

$$\begin{aligned} \ell(\theta) &= \frac{1}{\sqrt{n}} \|y - X\theta\|_2 + \lambda \|\theta\|_1 \\ Q(\theta, \theta_t) &= \ell(\theta_t) + \nabla \ell(\theta_t)'(\theta - \theta_t) + \frac{1}{2}(\theta - \theta_t)' \nabla^2 \ell(\theta_t)(\theta - \theta_t) \\ \theta_{t+0.5} &= \underset{\theta}{\text{Argmin}} Q(\theta, \theta_t) \text{ (coordinate descent)} \\ \theta_{t+1} &= \theta_t + \eta_t(\theta_{t+0.5} - \theta_t) \text{ (line search)} \end{aligned}$$

where the line search halves η_t until $\ell(\theta_{t+1}) \leq \ell(\theta_t)$.

The **proximal gradient** algorithm uses a coarser quadratic approximation

$$\ell(\theta) = \ell(\theta_t) + \nabla \ell(\theta_t)'(\theta - \theta_t) + \frac{1}{2} L \|\theta - \theta_t\|_2^2 + \lambda \|\theta\|_1$$

where L is obtained by line search.

Pathwise optimization, *i.e.*, a decreasing sequence of regularization parameters ending at the desired one, helps yield sparse solutions.

Two-player games for efficient non-convex constrained optimization
A. Cotter et al. (2018)

To solve a constrained, non-convex optimization problem, with non-differentiable (step function) constraints (e.g., in fair machine learning)

$$\begin{array}{ll} \text{Find} & \theta \\ \text{To minimize} & f(\theta) \\ \text{Such that} & \forall i \ g_i(\theta) \leq 0, \end{array}$$

consider the Lagrangian

$$\mathcal{L}(\theta, \lambda) = f(\theta) + \sum_i \lambda_i g_i(\theta) \quad (\lambda_i \geq 0)$$

and a differentiable proxy

$$\tilde{\mathcal{L}}(\theta, \lambda) = f(\theta) + \sum_i \lambda_i \tilde{g}_i(\theta)$$

and alternate

$$\begin{array}{l} \theta \leftarrow \underset{\theta}{\text{Argmin}} \ \tilde{\mathcal{L}}(\theta, \lambda) \\ \lambda \leftarrow \underset{\lambda \geq 0, \|\lambda\| \leq R}{\text{Argmin}} \ \mathcal{L}(\theta, \lambda) \end{array}$$

(we can compute the partial gradient $\partial \mathcal{L} / \partial \lambda$ of the original Lagrangian, even if the g_i 's are not differentiable: only $\partial \mathcal{L} / \partial \theta$ is problematic).

Keep the whole optimization history:

$$\mathbb{E}_{\theta \sim \text{Unif}(\{\theta_1, \dots, \theta_T\})} [g(\theta)]$$

is close to the minimum of g (there is no *pure* Nash equilibrium: all we can hope for is an approximation of a mixed Nash equilibrium).

(It is possible to reduce the number of points to $m+1$, where m is the number of constraints, with a simple linear program.)

Implementation in Tensorflow

Accelerated extra-gradient descent: a novel accelerated first-order method
J. Diakonikolas and L. Orecchia

Nesterov accelerated gradient descent can be interpreted as an explicit Euler discretization followed by a correction of the discretization error; instead, one can use an *implicit* Euler discretization.

Learned primal-dual reconstruction
J. Adler and O. Öktem

In the primal-dual optimization used to solve ill-posed problems (such as tomography) with a regularizing prior (total variation, TV), replace the proximal operator with a learned reconstruction operator, thereby combining deep-learning and model-based reconstruction.

Bridging the gap between constant step size stochastic gradient descent and Markov chains
A. Dieuleveut et al.

Averaged (Cesàro) constant-step-size stochastic gradient descent (SGD, Robins-Monro) benefits from Richardson-Romberg extrapolation.

Provable certificates for adversarial examples: fitting a ball in the union of polytopes
M. Jordan et al.

The *pointwise robustness* of a classifier is the radius of the largest ℓ^p ball centered at an input point x_0 for which the model output remains unchanged.

The *Chebyshev center* of a polytope is the center of the largest ℓ^p ball fitting inside it.

Spectral normalization for generative adversarial networks
T. Miyato et al. (2018)

Ensure the gradient of the discriminator remains bounded by normalizing each layer: divide each weight matrix W by its largest singular value (spectral norm, $\sigma(W)$); the Lipschitz constant of the nonlinearities is often 1.

Face aging with conditional generative adversarial networks
G. Antipov et al.

- Train a conditional GAN,
 - (age category, noise) \mapsto image
 - (they use 6 age categories);
- Train an auto-encoder to learn the inverse mapping
 - (age, latent) \mapsto image \mapsto (age, latent)
 - minimizing the Euclidean distance between the latent representations;
- Improve the latent representation by minimizing the distance between the embedding of the original and reconstructed image for a face recognition net (FaceNet);
- Change the age and re-generate the image.

Boosted generative models
A. Grover and S. Ermon

Boosting (multiplicative rather than additive) applied to the generator and/or the discriminator of a GAN.

GANSynth: adversarial neural audio synthesis
J. Engel et al. (2019)

Divide time into frames and generate, for each of them, a spectrogram, with log-amplitudes and (unwrapped) phases (the phase is needed to ensure a continuous signal; the unwrapped phase is smooth); train on the NSynth dataset (musical notes).

***Harmonizing maximum likelihood with GANs
for multimodal conditional generation***
S. Lee et al. (2019)

Conditional GANs are often trained with a loss combining minimax

$$\min_G \max_D \mathbb{E}_{x,y \sim \text{data}} \log D(x,y) + \mathbb{E}_{\substack{x \sim \text{label} \\ y \sim \text{noise}}} \log(1 - D(x, G(x,z)))$$

and reconstruction

$$\mathbb{E}_{\substack{x,y \sim \text{data} \\ z \sim \text{noise}}} \|y - G(x,z)\|_p^p$$

which worsens mode collapse.

Replace the reconstruction loss with a **moment reconstruction loss**: have the generator generate a *set* of samples, compute its moments μ, σ , and use a Gaussian (or Laplace) MLE loss

$$\mathbb{E} \left[\frac{(y - \mu)^2}{\sigma^2} + \frac{1}{2} \log \sigma^2 \right].$$

***Adversarial examples are not bugs,
they are features***
A. Ilyas et al.

The *usefulness* and *robustness* of a features f are

$$\begin{aligned} \rho(f) &= \mathbb{E}_{x,y} \text{Cor}(y, f(x)) \\ \gamma(f) &= \mathbb{E}_{x,y} \text{Cor}(y, f(x + \delta_x)) \end{aligned}$$

where $\delta)x$ is an adversarially chosen perturbation,

$$\delta_x = \underset{\delta \in \Delta}{\text{Argmin}} y \cdot f(x + \delta)$$

and $y \in \{0, 1\}$.

A robust model can be trained by minimizing the dver-sarial loss

$$\mathbb{E}_{x,y} \max_{\delta \in \Delta} \text{Loss}(x + \delta, y)$$

instead of $\mathbb{E}_{x,y} \text{Loss}(x, y)$.

Build a dataset of similar images using only useful robust features

$$x \mapsto x_r \in \underset{\delta \in \Delta}{\text{Argmin}} \|g(x_r) - g(x)\|$$

where g is the penultimate layer f a robust model and the optimization starts with noise.

Build a dataset of similar images using only useful non-robust features

$$x \mapsto x_a = \underset{x' : \|x' - x\| \leq \varepsilon}{\text{Argmin}} \text{Loss}(x', y')$$

where the loss is non-robust and the new class y' is chosen at random.

Non-robust models trained on those datasets have good performance: the existence of adversarial examples is a property of the dataset, not of the model – adversarial examples often transfer to other, independently-trained models.

***The Riemannian geometry
of deep generative models***
H. Shao et al.

Stochastic gradient descent can compute (discretized) geodesics (and parallel transport) of the data manifold of variational autoencoders (VAE) (the gradient of the decoder is expensive to compute, but that of the encoder suffices): it appears flat.

***Optimizing the latent space
of generative networks***
P. Bojanowski et al.

Learn a GAN without a discriminator (or an autoencoder with an unconstrained encoder):

$$\underset{\theta}{\text{Argmin}} \text{Mean}_{x \in \text{Data}} \min_{z \in \text{Ball}} \text{Loss}(g_\theta(z), x).$$

To avoid blurry images, use the *Laplacian pyramid* Lap₁ loss.

***Good semi-supervised learning
that requires a bad GAN***
Z. Dai et al. (2017)

GANs can be used for semi-supervised learning: the discriminator tries to forecast the class the sample comes from for labeled or unlabeled data, and a $(k + 1)$ st class for generated samples – the objective function has three terms, labeled, unlabeled and generated.

$$\begin{aligned} &\mathbb{E}_{(x,y) \sim \text{labeled}} \log P_D[y|x, y \leq k] + \\ &\mathbb{E}_{x \sim \text{unlabeled}} \log P_D[y \leq k|x] + \\ &\mathbb{E}_{x \sim P_G} \log P_D[y = k + 1|x] \end{aligned}$$

With a perfect generator, the GAN performs no better than supervised learning. Instead, use a discriminator of the form $P_D(k|X) \propto \exp(w_k f(x))$ for some features f , with support in $\{z : \forall k p_k(z) \leq \varepsilon_k\}$ where p_k is the distribution, on feature space, of class k .

***Approximation and convergence properties
of generative adversarial learning***
S. Liu et al. (2017)

GAN-style algorithms are of the form

$$\inf_{\nu} \sup_{f \in \mathcal{F}} \mathbb{E}_{\substack{x \sim \mu \\ y \sim \nu}} [f(x, y)],$$

i.e., $\inf_{\nu} \tau(\mu \| \nu)$ for an *adversarial divergence* on X

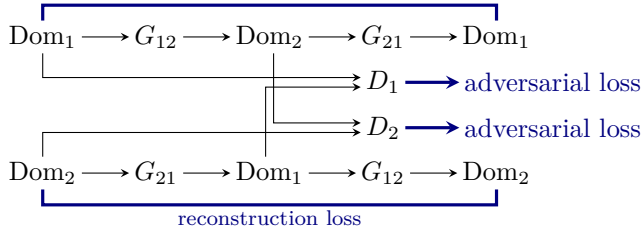
$$\tau(\mu \| \nu) = \sup_{f \in \mathcal{F}} \mathbb{E}_{\mu \otimes \nu} [f],$$

for some $\mathcal{F} \subset \mathcal{C}_b(X^2)$.

For limited-capacity discriminators, the set of distributions μ^* minimizing $\tau(\mu \| \mu^*)$ is the set of distributions with generalized moments (depending on τ) matching those of μ .

***Learning to discover cross-domain relations
with generative adversarial networks***
T. Kim et al.

To convert images from one domain to another (e.g., bags to shoes), without paired data:



The measure of intelligence
F. Chollet

Intelligence is not “skill”, but “skill-acquisition efficiency”: the ability to turn prior knowledge and training data into skill.

For humans, priors (*Core Knowledge*) include objectness (elementary physics: the world is made of objects), agentness (goal-directedness), numbers and geometry.

Intelligence could be defined, for a fixed level of skill, as

$$\text{Intelligence} = \frac{\text{Skill} \times \text{Generalization Difficulty}}{\text{Prior} \times \text{Experience}},$$

averaged over training datasets (curriculum) and tasks. The factors can be defined from information theory (Kolmogorov complexity).

$$\text{Difficulty} = \frac{H(\text{Solution} | \text{Optimal solution})}{H(\text{Solution})}$$

$$\text{Prior} = 1 - \frac{H(\text{Solution})}{H(\text{Solution} | \text{Initial State})}$$

$$\text{Experience} = H(\text{Solution} | \text{IS}) - H(\text{Solution} | \text{Data}).$$

The abstraction and reasoning corpus (ARC) contains 1000 tasks, each with 3 or 4 examples, one test, and allows for 3 trials. The task is to complete a grid, using geometric principles not explicitly stated, from a few examples. Humans can solve the problems without any training – the geometric prior suffices.

Alternatives include video games (train on a level and test on others, train on a game and test on other similar games) or data science competitions (train on a few Kaggle dataset, test on others).

***Universal intelligence:
a definition of machine intelligence***
S. Legg and M. Hutter (2007)

Define the intelligence of a reinforcement learning algorithm π as its expected value,

$$\Upsilon(\pi) = \sum_{\mu \in E} 2^{-K(\mu)} V_{\mu}^{\pi},$$

where E is the set of environments to test and $K(\mu)$ the Kolmogorov complexity (approximated by the minimum description length and the running time) of environment μ (simple environments are more probable)

***Deep lattice networks
and partial monotonic functions***
S. You et al (2017)

Lattices (interpolated lookup tables, which can easily enforce monotonicity constraints) and calibrators (1-dimensional lattices) can be used as layers and activation functions.

***Scaling limits of wide neural networks
with weight sharing: Gaussian process
behaviour, gradient independence
and neural tangent kernel derivation***
G. Yang

Infinitely large, 1-layer neural nets with random weights are Gaussian processes.

***Identity crisis:
memorization and generalization
under extreme overparametrization***
C. Zhang

When asked to reconstruct a *single* example,

- Some networks (FC) learn the constant function (memorization);
- Some (shallow CNN) learn the identity function (generalization).

***Spectrally-normalized margin bounds
for neural networks***
P.L. Bartlett et al.

The *spectral complexity* of a network is

$$R = \left(\prod \rho_i \|A_i\|_{\sigma} \right) \left(\sum \frac{\|A'_i - M'_i\|_{2,1}^{2/3}}{\|A_i\|_{\sigma}^{2/3}} \right)^{3/2}$$

where ρ_i is the Lipschitz constant of the nonlinearity of layer i , $\|A_i\|_{\sigma}$ is the spectral norm of the weight matrix A_i , M_i is a reference matrix (e.g., $M_i = 0$, or $M_i = I$ for ResNets), $\|A\|_{p,q}$ is the (p, q) matrix norm

$$\left\| (\|A_{\cdot 1}\|_p, \dots, \|A_{\cdot m}\|_p) \right\|_q.$$

It is lower for networks that generalize better. For multiclass classification, we expect the normalized margin

$$\frac{f(x)_y - \max_{i \neq y} f(x)_i}{R \cdot \|x\|_2 / n}$$

to increase as generalization improves.

***Not all samples are created equal:
deep learning with importance sampling***
A. Katharopoulos and F. Fleuret

When training a neural net, *importance sampling* helps focus the computations on informative examples, but the importance, the per-sample gradient, is expensive to compute – try $\|\nabla_{x_i^L} \text{Loss}\|$ instead, where i is the sample and x^L the activation of the last layer.

***Deep, skinny neural networks
are not universal approximators***
J. Johnson (2019)

Skinny networks (the dimension of the hidden layers never exceeds that of the input) cannot approximate functions with a level set containing a bounded connected component.

***PaperRobot:
incremental draft generation of scientific ideas***
Q. Wang et al.

Extract entities (gene, disease, chemical) and relations (marker, therapeutic, increased expression, etc.) from research papers to build a knowledge graph (KG); use both graph structure and contextual (text) embeddings of the nodes to forecast missing links; use them to automatically write papers: entities+relations \rightarrow title \rightarrow abstract \rightarrow conclusion and future work \rightarrow next article \rightarrow etc.

Attention is not explanation
S. Jain and B.C. Wallace

Attention weights need not be correlated with importance measures: it is possible to change them to meaningless ones without impacting performance.

Are sixteen heads really better than one?
P. Michel et al.

Multiple attention heads (in transformer networks, e.g., BERT) are useful during training but most layers can be pruned to a single head without a significant drop in performance.

***Fair is better than sensational:
man is to doctor as woman is to doctor***
M. Nissim et al. (2019)

The code used to compute analogies of the form “doctor + woman – man = nurse” excludes one of the inputs (“doctor”) and even similar items (“gynecologist”, etc.).

***A kernel for time series
based on global alignments***
M. Cuturi et al. (2006)

Dynamic time-warp (DTW), Smith-Waterman local alignments and edit distance do not readily lead to a positive definite kernel.

Gaussian and polynomial kernels are not satisfactory either: they cannot deal with time series of different lengths and ignore the time series structure.

Taking the best alignment (with no gaps but possible repetitions: gaps make more sense for biological sequences, repetitions for time series, e.g., speech) does not give a positive definite kernel, but combining them all often does:

$$\begin{aligned} S(\pi) &= \text{score of alignment } \pi \\ \kappa(x, y) &= \sum_{\pi} e^{S(\pi)} \\ &= \sum_{\pi} \exp \sum_i \phi(x_{\pi_1(i)}, y_{\pi_2(i)}) \\ &= \sum_{\pi} \prod_i \exp \phi(x_{\pi_1(i)}, y_{\pi_2(i)}). \end{aligned}$$

The kernel can be computed in $|x| \cdot |y|$ iterations by dynamic programming (as for DTW).

Natural language understanding
C. Potts et al. (2019)

Vector space models require several choices:

- Design matrix: word \times document, word \times word, word \times context, etc.;
- Reweighting, to “de-emphasize the mundane and the quirky”: probabilities, length normalization, TF-IDF, PMI, PPMI, T-test (prefer PMI);

$$\text{PMI} = \log \frac{\text{observed}}{\text{expected}} = \log \frac{X_{ij}}{X_{i.} X_{.j} / X_{..}}$$

- Dimension reduction: PCA, LSA, PLSA, LDA, NMF, etc.
- Vector comparison: Euclidean, cosine, Dice, Jaccard, KL, etc. (prefer cosine).

Large, flat windows capture semantic information, small, scaled windows syntactic (collocational) information.

GloVe learns word vectors such that

$$\langle u | v \rangle \propto P(u, v);$$

more precisely, they minimize

$$\sum_{ij} f(X_{ij})(w'_i \tilde{w}_k + b_i + \tilde{b}_k - \log X_{ik})$$

where $f(x) = \text{Min}\{(x/x_{\max})^\alpha, 1\}$, $x_{\max} = 100$, $\alpha = .75$.

Word2Vec learns word embeddings such that

$$P(b|a) \propto \exp(w_a w_b).$$

Maximizing the likelihood is problematic because of the denominator,

$$\text{Argmax}_{x,w} \sum_{a \sim b} \log \frac{\exp(x_a w_b)}{\sum_c \exp(x_a w_c)}.$$

Instead, use *noise contrastive estimation*

$$\sum_{a \sim b} -\log \sigma(x_a w_b) + \sum_{a, b} \log \sigma(x_a w_b).$$

Retrofitting modifies the word vectors to incorporate information (e.g., sentiment, WordNet) from a *knowledge graph*.

$$\text{Minimize} \sum_q \|q_i - \hat{q}_i\|^2 + \sum_{(i,j) \in E} \beta_{ij} \|q_i - q_j\|^2$$

To compare models, use the Wilcoxon signed rank test, if you run several experiments, or the *McNemar*, to compare confusion matrices, if you cannot.

Features for sentiment analysis could include: bag of subparts (subtrees with at most k leaves), negation (add `_NEG` to all the tokens until the end of the subtree – idem for other modifiers: *might*, etc.), lexicon-derived features, modal adverbs (totally, quite possibly, etc.), non-literal language, character n -grams, tf-idf, part-of-speech, sentence length, GloVe, etc. Use BERT (pretrained and fine-tuned) or a BiLSTM with subtree labels, or manual features

Distant supervision builds training data for the relation extraction problem from a knowledge base (database of known relations), assuming that, for each known relation, any sentence mentioning both entities expresses that relation.

For natural language inference (entailment), try the following features: words that appear in both clauses, words that appear in either clause, wordnet features (pairs of words, in the first and second clause, with $w_1 \succ w_2$ or $w_1 \preceq w_2$, edit distance, negation, models, quantifiers (every, some), named entities.

Given the hidden states for the words in the first, resp. second, clause, h_1, h_2, h_3 , resp. h_A, h_B, h_C , the **attention mechanism** is

Scores	$\tilde{\alpha} = [h'_C h_1; h'_C h_2; h'_C h_3]$
Attention	$\alpha = \text{softmax } \tilde{\alpha}$
Context	$\kappa = \text{mean}(\alpha_1 h_1, \alpha_2 h_2, \alpha_3 h_3)$
Combination	$\tilde{h} = \tanh([\kappa; h_C]W)$
Classifier	$y = \text{softmax}(\tilde{h}W + b).$

Other score functions are possible: $h'_C h_i$, $h'_C W h_i$ or $W[h_C; h_i]$.

Language understanding requires *grounding*: many utterances depend on the context (“there aren’t any”), so do answers (“Where are you from?”) and even modals (is “The door must be open” a rule or a deduction?).

In the **rational speech act** (RSA) model, the listener has a model of how the speaker thinks, and conversely.

$$\begin{aligned} \ell_0(\text{world}|\text{text}) &\propto \text{language}(\text{text}, \text{world}) \cdot P(\text{world}) \\ s_1(\text{text}|\text{world}) &\propto \left(\frac{\ell_0}{C(\text{word})} \right)^\lambda \\ \ell_1(\text{world}|\text{text}) &\propto s_1(\text{text}|\text{world}) P(\text{world}) \end{aligned}$$

literal listener = lexicon \times prior

pragmatic speaker = literal speaker – message costs

pragmatic listener = pragmatic speaker \times prior

Semantic parsing transforms text into a formal language, suitable for constraint satisfaction solvers.

The F_β evaluation metric is a weighted harmonic average of recall (weight β^2) and precision (weight 1); use $\beta = 1$ or 0.5; for multiclass classification, the *macro* F_β is the (unweighted) average of the F_β ’s for each class.

Check if your model performs well in an adversarial setting:

- Add a misleading sentence to the input;
- Replace a word with synonym, a more general word, a comparable word (co-hyponym);
- Swap subject and object;
- Move an adjective to another noun phrase;
- Add negations; add a negation and swap subject and object (contrapositive).

Easy data augmentation uses simple changes:

- Synonym replacement;
- Random insertion: take a word in the sentence and put another copy of it somewhere else;
- Random swap;
- Random deletion.

Back-translation also provides data augmentation.

Language models provide *contextual embeddings*:

- ELMo, a biLSTM with highway layers to forecast the next word
- Transformers use multi-headed self-attention, position embedding, dropout, residual connections and layer normalization
- BERT is a masked language model, with a small vocabulary (subwords), position embedding, using CNNs with several perceptive fields and max pooling.

To probe black-box models, take the word or sentence representation and check if you can predict, with a shallow network: sentence length, which word is present, tense, POS, named entities, semantic information (anything from WordNet, sentiment, entailment, etc.), tense. Alternatively, change something in the sentence (tense, word order by swapping two consecutive words) and see how the representation changes.

Linear algebraic structure of word senses, with applications to polysemy S. Arora et al.

Use *sparse coding* (and alternating minimization) to decompose a given word embedding $w \mapsto v_w$ into disambiguated embeddings, $\alpha_{w,i} A_i$,

$$v_w = \sum_i \alpha_{w,i} A_i + \text{noise}$$

where the $\alpha_{w,\cdot}$ are sparse; the A_i ’s are the contexts (“atoms of discourse”).

**Contextual string embeddings
for sequence labelings**
A. Akbik et al.

A character-level (BiLSTM) language model provides *contextualized* word embeddings (somehow aggregate the hidden states corresponding to the word letters), which can be fed to another BiLSTM for named entity recognition (NER) or some other labeling task.

**Machine translation
with weakly paired bilingual documents**
(2019)

Use weakly-aligned documents (e.g., matching Wikipedia articles) to train a machine translation model: find matching sentences from the cosine similarity of their bag-of-word embeddings, from a multilingual word embedding (MUSE).

**A syntactic neural model
for general-purpose code generation**
P. Yin and G. Neubig

When translating natural language into code, do not directly use a sequence-to-sequence model: translate into an AST (abstract syntax tree) instead, to ensure the code is syntactically correct.

Data from manually-annotated Django code, a card game, and an automation app.

**Neural machine translation
and sequence-to-sequence models: a tutorial**
G. Neubig (2017)

Tutorial, with exercises, covering count-based n -gram language models

$$P[e_{1:n}] = \prod_i P[e_i | e_{1:i-1}]$$

$$\approx \prod_i P[e_i | e_{i-k:i-1}],$$

with modified Kneser-Ney smoothing, log-linear models (with features ϕ and a softmax)

$$P[e_{1:n}] \approx \prod_i P[e_i | \phi(e_{i-k:i-1})],$$

neural networks (idem, but non-linear), RNNs, Seq2seq and attention.

**A regularized framework for sparse
and structured neural attention**
V. Niculae and M. Blondel

The maximum

$$\text{Max}^* y = \begin{cases} 0 & \text{if } y \in \Delta \\ \infty & \text{otherwise} \end{cases}$$

$$\text{Max } x = \text{Max}^{**} x = \sup_{y \in \Delta} y'x$$

can be regularized

$$\text{Max}_\Omega^* y = \begin{cases} \gamma \Omega(y) & \text{if } y \in \Delta \\ \infty & \text{otherwise} \end{cases}$$

$$\text{Max}_\Omega x = \text{Max}_\Omega^{**} x = \sup_{y \in \Delta} y'x - \gamma \Omega(y).$$

The softmax can be generalized to

$$\Pi_\Omega(x) = \underset{y \in \Delta}{\text{Argmax}} y'x - \gamma \Omega(y) = \nabla \text{Max}_\Omega(x).$$

The softmax attention mechanism yields dense weights: sparsemax gives sparse weights, fusedmax sparse continuous weights, oscarmax clustered weights regardless of position.

$$\Omega(y) = \sum y_i \log y_i \quad \text{negentropy}$$

$$\Pi_\Omega(x) = \frac{\exp(x/\gamma)}{\sum \exp(x_i/\gamma)} \quad \text{softmax}$$

$$\Omega(y) = \frac{1}{2} \|y\|_2^2$$

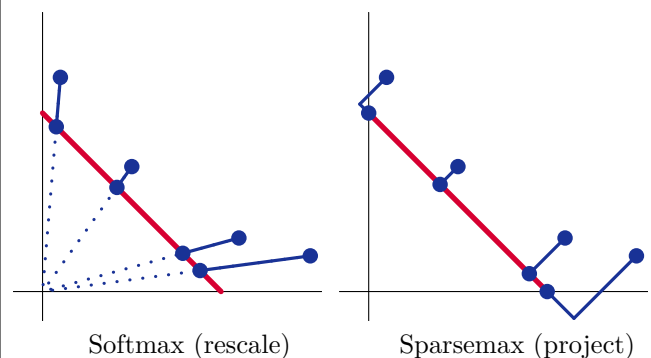
$$\Pi_\Omega(x) = \underset{y \in \Delta}{\text{Argmin}} \|y - x/\gamma\|^2 \quad \text{spacemax}$$

$$\Omega(y) = \frac{1}{2} \|y\|_2^2 + \lambda \sum \|y_{i+1} - y_i\| \quad \text{fused lasso}$$

$$\Pi_\Omega(x) = \underset{y \in \Delta}{\text{Argmin}} \|y - x/\gamma\|^2 + \lambda \sum \|y_{i+1} - y_i\|$$

$$\Omega(y) = \frac{1}{2} \|y\|_2^2 + \lambda \sum_{i < j} \text{Max}\{\|y_i\|, \|y_j\|\}$$

$$\Pi_\Omega(x) = \underset{y \in \Delta}{\text{Argmin}} \|y - x/\gamma\|^2 + \lambda \sum_{i < j} \text{Max}\{\|y_i\|, \|y_j\|\}$$



Searching for activation functions
P. Ramachandran et al.

Automated search for new activation functions, by combining 30+ common unary and binary functions, using an RNN trained with RL to explore the search space, suggests $\text{swish}(x) = x \cdot \text{sigmoid}(x)$.

Learning explanatory rules from noisy data
R. Evans and E. Grefenstette (2017)

Inductive logic programming (ILP) turns a set of positive and negative examples, e.g.,

$$P = \{0, 2, 4, 6, 8, \dots\}$$

$$N = \{1, 3, 5, 7, 9, \dots\}$$

into rules, e.g.,

$$\begin{aligned}\text{even}(X) &\leftarrow \text{zero}(X) \\ \text{even}(X) &\leftarrow \text{even}(Y) \wedge \text{succ2}(Y, X) \\ \text{succ2}(X, Y) &\leftarrow \text{succ}(X, Z) \wedge \text{succ}(Z, Y).\end{aligned}$$

The consequences of a set of rules can be obtained by breadth-first search (*forward chaining*).

The top-down approach to ILP, which generates programs and tests them on the positive and negative examples, can be turned into a satisfiability problem, and relaxed into a continuous one (replace boolean values with number in $[0, 1]$), amenable to gradient descent (forward chaining is differentiable: only perform T steps of it).

***Image search using multilingual texts:
a cross-modal learning approach
between image and text***
M. Portaz et al.

Embed text (MUSE, a multilingual FastText extension) and images in the same latent space.

***Deep learning for image super-resolution:
a survey***
Z. Wang et al.

The different approaches include pre-upsampling ($\text{up} \cdot \text{conv}^k$), post-upsampling ($\text{conv}^k \cdot \text{up}$), progressive upsampling ($((\text{conv}^k \cdot \text{up})^n)$), iterative up-and-down-sampling ($((\text{conv} \cdot \text{up} \cdot \text{down})^n \cdot \text{up})$).

The upsampling can be nearest-neighbour, linear, bilinear, or a fractionally-strided convolution (“deconvolution”).

The network can have residual connections, dense connections, and parallel paths (as in inception modules).

Loss functions include pixel loss, content loss (hidden layer of some classifier), texture loss (Gram matrix of such a layer, as in style transfer), adversarial loss, cycle-consistency loss.

***EfficientNet: rethinking model scaling
for convolutional neural networks***
M. Tan and Q.V. Le (2019)

Scale your network as

$$\begin{array}{ll}\text{depth} & d = \alpha^\phi \\ \text{width} & w = \beta^\phi \\ \text{resolution} & r = \gamma^\phi\end{array}$$

where α, β, γ are determined by grid search and satisfy $\alpha^2 \beta^2 \gamma^2 = 2$, $\alpha, \beta, \gamma \geq 1$.

***Interactive sketch & fill: multiclass
sketch-to-image translation***
A. Ghosh et al. (2019)

GAN to convert partial sketches to complete sketches, and then complete sketches to images. The desired object class is enforced using channel-wise gates – conditional GANs tend to forget the input class. The data comes from occluded contours, automatically extracted from images.

***Measuring the tendency of CNNs to learn
surface statistical regularities***
J. Jo and Y. Bengio

CNNs to not recognize high-level features, but focus on statistical regularities: for instance, they are not robust to Fourier filtering.

***Deep normal estimation for automatic shading
of hand-drawn characters***
M. Hudon et al.

Use a CNN (U-net) to infer normals from (and automatically shade) hand-drawn 2D characters; training data from Blender’s *FeeStyle* (an edge- and line-based non-photorealistic (NPR) rendering engine).

Deformable convolutional networks
J. Dai et al.

Deformable convolution networks use deformed grids as filters (like *active convolution*), whose offsets are position-dependent, computed from the features of the previous layer.

When a neural net generates both a list of regions of interest (ROI) and (localized) features, *ROI pooling* pools the features in each ROI (the pooling regions do not have a fixed size), usually dividing the ROI into a regular grid. It can be deformed: deformable ROI pooling.

***In defence of the triplet loss
for person re-identification***
A. Hermans et al.

Person re-identification often uses the triplet loss

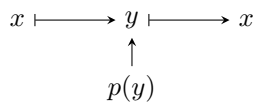
$$\sum_{\substack{i,j,k \\ y_i=y_j \\ y_i \neq y_k}} (m + d_{ij} - d_{ik})_+,$$

but since the number of triplets grows quadratically, *hard triplet mining* is necessary. Instead,

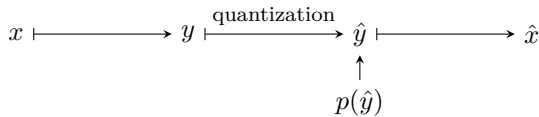
- Build a batch of P classes, with K images in each, and compute the loss for all resulting triplets, or only for the hardest positive and hardest negative for each anchor;
- Use the (non-squared) Euclidean distance;
- Use a soft margin (softplus).

***Variational image compression
with a scale hyperprior***
J. Ballé et al. (2018)

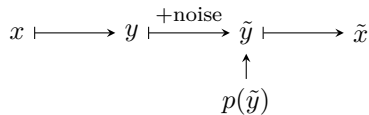
Variational autoencoders (VAE)



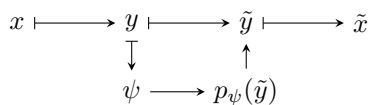
can be used for image compression



and trained by replacing the quantization with uniform additive noise



Have the variational distribution $p(\tilde{y})$ depend on the image x (or on the latent representation y).



SegNet: a deep convolutional encoder-decoder architecture for image segmentation
V. Badrinarayanan et al. (2016)

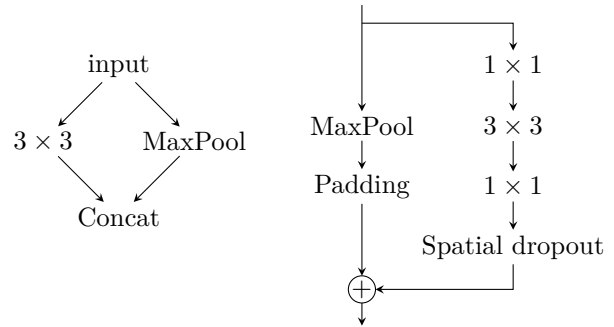
SegNet performs pixel-wise image segmentation by using

- An encoder (VGG16);
- Followed by a decoder, which upsamples the features using the *pooling indices* from the encoder’s pooling layers;
- Followed by a pixelwise classification layer.

ENet: a deep neural network architecture for real-time semantic segmentation
A. Paszke et al. (2016)

The EfficientNet (ENet) combines the following ideas to reduce the number of floating point operations when compared with VGG16: SegNet (pooling indices), early downsampling, decoder smaller than the encoder, *PReLU* instead of ReLU, pooling and convolution in parallel, *factorizing filters* (5×1 and 1×5 instead of 5×5), dilated convolutions, *spatial dropout*

(dropout all channels at random locations).



ShuffleNet: an extremely efficient convolutional neural network for mobile devices
X. Zhang et al. (2017)

The 1×1 convolutions are dense matrix multiplications: they are costly and can be replaced with block-diagonal matrices (“pointwise group convolutions”) and (random) permutation matrices (“channel shuffle”).

ShuffleSeg: real-time semantic segmentation network
M. Gamal et al. (2018)

RepPoint point set representation for object detection
Z. Yang et al. (2019)

Rectangular bounding boxes do not account for an object’s shape and pose. Instead, use a set of (9) points, inside the object, computed with a feature pyramidal network (FPN) with deformable convolutions, trained with a localization loss (it should give the same bounding box) and a recognition loss.

Complex-YOLO: an Euler-region-proposal for real-time 3d object detection on point clouds
M. Simon et al. (2019)

To feed Lidar data to a YOLO network to find 3D bounding boxes, discretize the cloud of points into a top-view image, with 3 channels: maximum height, maximum intensity, number of points. For the orientation of the boxes, output a complex number of modulus 2 instead of an angle to avoid singularities.

Oriented response networks
Y. Zhou et al.

To detect rotation-invariant patterns, and keep track of their orientation, consider N rotations of a given filter (sharing the same parameters).

**Limitations
of the empirical Fisher approximation**
F. Kunstner et al.

The *empirical Fisher*

$$\sum_n \nabla_{\theta} \log p_{\theta}(y_n|x_n) \nabla_{\theta} \log p_{\theta}(y_n|x_n)'$$

is not a good approximation of the Fisher information matrix

$$\sum_n \mathbb{E}_{y \sim p_{\theta}(\cdot|x_n)} [\nabla_{\theta} \log p_{\theta}(y_n|x_n) \nabla_{\theta} \log p_{\theta}(y_n|x_n)']$$

Do not use it for natural gradient descent [but it is still better than plain gradient descent].

**Model compression
by entropy penalized reparametrization**
D. Oktay et al.

To compress the weights of a neural net, **scalar quantization** discretizes each coordinate independently, while **vector quantization** uses some clustering algorithm (e.g., *k*-means).

Learn another network to generate weights from a low-dimensional, discrete (scalar quantization) latent representation (hypernetwork) with a penalty for the length of the Shannon-encoded latent representation (entropy).

**Learning relational representations
with auto-encodding logic programs**
S. Dumančić et al.

An auto-encoder, with (Prolog) facts as input and latent representation, and (inverse) logical (Prolog) programs as encoder and decoder can be learnt using constraint satisfaction tools (large neighbourhood search: BUSL learner for Markov logic networks (MLN)), minimizing the reconstruction error with constraints on

- The average number of facts per predicate;
- Redundancy (if $c_1 \models c_2$, include at most one);
- Coverage (reconstruct at least one of each predicate).

**Joint semantic analysis and
morphological analysis of the derived word**
R. Cotterell and H. Schütze

Estimate word embeddings by modeling the derivation of words, e.g.,

questionably = question:stem + able:suffix + ly:suffix.

Relational knowledge distillation
W. Park et al.

To distill a (teacher) model (into a student model), do not just try to reproduce the individual outputs, but also the distances and angles.

**Interacting conceptual spaces I:
grammatical composition of concepts**
J. Bolt (2016)

Many grammars can be modeled as a monoidal category \mathfrak{G} , with reduction rules (e.g., noun \otimes verb \otimes noun \rightarrow sentence) as morphisms. To model the meaning of sentences from that of words, learn:

- A word embedding Words $\rightarrow V$;
- A monoidal functor $\mathfrak{G} \rightarrow \mathfrak{Vect}$.

The category of vector spaces can be replaced by that of convex sets and relations (more generally: convex algebras and convex relations).

Byzantine-tolerant machine learning
P. Blanchard et al.

Distributed stochastic gradient descent is not tolerant to Byzantine failure: use the vector closest to the average of its $n - k$ neighbours.

**Solving ill-posed inverse problems
using iterative deep neural networks**
J. Adler and O. Öktem (2017)

Use deep neural networks to find an approximate inverse of $T : X \rightarrow Y$, minimizing

$$\mathbb{E}_{\substack{x \sim \text{data} \\ y = Tx}} [\text{Loss}(TT_{\theta}^{-1}y, y) + \text{Penalty}(T_{\theta}^{-1}y)].$$

Snapshot ensembles: train 1, get M for free
G. Hunag et al. (2017)

Use a cyclic learning rate and ensemble the local minima found during the optimization.

Personalized re-ranking for recommendation
C. Pei et al. (2019)

Reranking models, to transform a list of search results, sorted by their closeness to a query (or the estimated probability that the user will click on it), should be reranked to account for interaction (e.g., to ensure diversity and avoid redundant results). The reranking model takes, as input, features of the items, their positions, possibly user-specific item features (a learned user-specific linear transformation of the item features). It is often an RNN generating a sequence of scores, along which to sort the items, but it can be replaced with a transformer with attention.

**FreezeOut: accelerate training
by progressively freezing layers**
A. Brock et al.

Freeze the first layers of the network after a while: they converge faster, to simple configurations (e.g., edge detectors).

Stochastic depth, in a resnet, drops whole layers at a time.

On connected sublevel sets in deep learning
Q. Nguyen (2019)

If there are enough hidden neurons (half as many in the first hidden layer as training samples), then the sublevel sets of the loss function are connected – in particular, the set of global minima is connected.

**A constructive approach
for one-shot training of neural networks
using hypercube-based topological coverings**
W.B. Daniel and E. Yeung

Decision trees are special cases of neural nets (and can be used to initialize them or choose their topology): decision boundaries are ReLU operations.

**Anchors:
high-precision model-agnostic explanations**
M.T. Ribeiro et al. (2018)

To explain a model forecast, find a single rule of the form $a_1 \wedge \dots \wedge a_k$, with beam search, which applies to the observation of interest (local), but with high (global) precision and decent coverage.

Contrary to purely local explanations (LIME), anchors give correct forecasts with high probability whenever they apply.

Axiomatic attribution for deep neural networks
M. Sundararajan et al. (2017)

Attribution of the output of a neural net to input features should be

- Sensitive: if x_1 and x_2 only differ by one feature and $f(x_1) \neq f(x_2)$, then this feature's contribution should be non-zero;
- Implementation-invariant: two different neural nets computing the same function should have the same contributions.

Integrated gradients (along a rectilinear path, between a baseline input, e.g., a black image, and the image of interest) satisfy those conditions.

**BOHB: robust and efficient
hyperparameter optimization at scale**
S. Falkner et al. (2018)

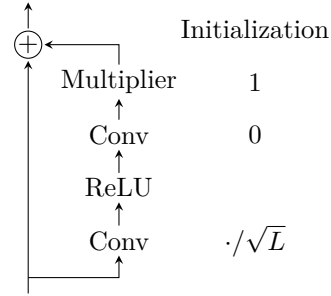
Hyperband uses *successive halving* for hyperparameter tuning:

- Pick N configurations at random;
- Train them for a while;
- Discard the worst half;
- Train the remaining models for a while;
- Continue until only one is left.

BOHB uses Bayesian optimization to replace the configurations at the beginning of each HB iteration.

**Fixup initialization:
residual learning without normalization**
H. Zhang et al. (2019)

Initialize the classification layer, the last layer of the residual branches, and the biases to zero; rescale the other residual layers by \sqrt{L} , where L is the number of layers (or $L^{1/2(m-1)}$, if there are m layers in the residual blocks).



**Geometric matrix completion
with recurrent multi-graph networks**
F. Monti et al. (2017)

Matrix completion is often done with a low-rank approximation, or its relaxation with the nuclear norm (sum of the singular values)

$$\begin{aligned} & \underset{X}{\text{Minimize}} \text{rank}(X) + \lambda \|\Omega \odot (X - Y)\|_F^2 \\ & \underset{X}{\text{Minimize}} \|X\|_* + \lambda \|\Omega \odot (X - Y)\|_F^2. \end{aligned}$$

Given a user-item matrix, consider the corresponding row (user) and column (item) similarity graphs, and their Laplacians

$$\begin{aligned} \Delta_r &= \Phi_r \Lambda_r \Phi_r' \\ \Delta_c &= \Phi_c \Lambda_c \Phi_c'. \end{aligned}$$

The *geometric completion* problem is

$$\underset{X}{\text{Minimize}} \|X\|_{\Delta_r}^2 + \|X\|_{\Delta_c}^2 + \lambda \|\Omega \odot (X - Y)\|_F^2$$

where the **Dirichlet norm** is $\|X\|_{\Delta} = \text{trace}(X' \Delta X)$. Define the Fourier transform and the convolution as

$$\begin{aligned} \hat{X} &= \Phi_r' X \Phi_c \\ X * Y &= \Phi_r (\hat{X} \odot \hat{Y}) \Phi_c' \end{aligned}$$

and approximate the filter Y with Chebychev polynomials

$$\tilde{X} = \sum_{0 \leq i, j \leq p} \theta_{ij} T_i(\tilde{\Delta}_r) X T_j(\tilde{\Delta}_c)$$

where $\tilde{\Delta} = 2\lambda_n^{-1} \Delta - I$ are the rescaled Laplacians, with eigenvalues in $[-1, 1]$.

Train such a *multi-graph CNN* (MGCNN) for geometric matrix completion, and feed its features to an RNN.

***Probabilistic matrix factorization
for automated machine learning***
N. Fusi et al.

AutoML can be framed as a recommendation problem, matching datasets (users) to ML pipelines (items), with matrix factorization to recommend pipelines, or with *probabilistic matrix factorization* to optimize an acquisition function (as in Bayesian optimization).

Probabilistic matrix factorization
R. Salakhutdinov and A. Mnih

Matrix factorization $R \approx U'V$ can be made Bayesian

$$\underset{U,V,\sigma}{\text{Argmax}} \prod_{\substack{i \text{ user} \\ j \text{ item} \\ i \text{ rated } j}} \phi(R_{ij}, \mu = U'_i V_j, \sigma^2)$$

***Graph2Seq: graph to sequence learning
with attention-based neural networks***
K. Xu et al.

Estimate a node embedding by

- Aggregating neighbour features (with mean, LSTM on random permutations of the neighbours, separating inbound and outbound neighbours, or element-wise max of a 1-layer neural net);
- Concatenate with the node features;
- Feed to a 1-layer neural net to get new node features;
- Iterate 3 to 5 times (with different aggregators and neural nets).

Convert it to a graph embedding with an elementwise max/min/mean, and feed this graph representation to a sequence generator with attention.

***CayleyNets: graph convolutional neural
networks with complex rational spectral filters***
R. Levie et al. (2017)

In the (spectral) graph convolution $f \mapsto \Phi g(\Lambda) \Phi' f$, where $\Delta = \Phi \Lambda \Phi'$ is the eigen decomposition of the Laplacian Δ and $f \mapsto \hat{f} = \Phi' f$ the Fourier transform, do not model g with Chebychev polynomials (a high degree is needed to produce narrow-band filters – in presence of communities, eigenvalues tend to cluster) but with Cayley “polynomials” $\lambda \mapsto \text{Re } C(h\lambda)^k$ where the Cayley transform is

$$\begin{cases} \mathbf{R} & \longrightarrow e^{i\mathbf{R}} \setminus \{1\} \\ \lambda & \longmapsto \frac{\lambda - i}{\lambda + i} \end{cases}$$

and h a zoom parameter.

***Amplifiers and suppressors of selection
for the Moran process in undirected graphs***
G. Giakkoupis (2018)

The **Moran process** models the spread of genetic mutations on a graph:

- Start with a single mutant node;
- Pick a node at random, with probability proportional to its fitness (1 for non-mutants, $r > 1$ for mutants);
- The node copies itself to one of its neighbours.

There exist families of (directed or undirected) graphs for which

- The probability of fixation tends to 1 (strong amplifier);
- The probability of extinction tends to 0 (strong suppressor).

***A unified framework
for structured graph learning
via spectral constraints***
S. Kumar et al.

Using the Laplacian, add constraints to the graphical lasso (for graph reconstruction from a covariance matrix) to make the graph bipartite, regular, or to specify the number of connected components.

***Multi-dimensional count sketch:
dimension reduction
that retains efficient tensor operations***
Y. Shi and A. Anandkumar

The **count sketch**

$$\text{CS}(x) = \left(\sum_{h(i)=j} s_i x_j \right)_j = (s \odot x)' H$$

can be generalized to tensors

$$\text{MS}(T) = ((s_1 \otimes \cdots \otimes s_p) \circ T)(H_1, \dots, H_p)$$

where $T \in \mathbf{R}^{n_1 \times \cdots \times n_p}$, $s_i \in \{\pm 1\}^{n_i}$, $H_i \in \{0, 1\}^{n_i \times m_i}$ with exactly one 1 in each row.

***Gorilla: a fast, scalable, in-memory
time series database***
T. Pelkonen et al. (2015)

In 2013, Facebook moved from an HBase-backed OpenTSDB-like time series database to an in-memory one (for the last 26 hours, HBase for the rest), using

- Delta of delta (variable length encoding) compression for timestamps: observations are almost equally spaced;
- XOR encoding (and VLE compression) for numeric values: nearby values are often close, and share the beginning of the mantissa, the exponent, and sometimes the precision.

Also check InfluxDB (no Hadoop required, distributed, metadata), OpenTSDB (HBase, metadata), Whisper (aka Graphite, regularly-spaced data, RDD, on-disk), M3DB (from Uber).

Towards understanding generalizations of deep learning: perspective of loss landscapes
L. Wu et al.

Minima whose bassins of attraction have a larger volume generalize better (and are more likely to be reached after random initialization).

Loss landscapes of regularized linear autoencoders
D. Kunin et al. (2019)

Linear autoencoders are GL_k -invariant: they learn the subspace spanned by the first k principal directions, but not the separate directions. L^2 -regularized linear encoders do.

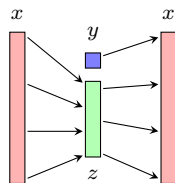
A PCA-like autoencoder
S. Ladjal et al. (2019)

To make the coordinates of the latent space of an autoencoder more interpretable and independent:

- Iteratively increase the dimension of the latent space, keeping the already learned dimensions fixed (discard and learn the decoder anew each time);
- Add a penalty for the (off-diagonal) covariances.

Fader networks: manipulating images by sliding attributes
G. Lample et al. (2018)

To train an autoencoder some of whose latent space coordinates have a pre-specified interpretation (e.g., presence of glasses, beard, etc.), add the interpretable feature to the latent space, and ensure it is not present in the rest of the latent representation with a discriminator $D : z \mapsto y$ and an adversarial loss.



Early visual concept learning with unsupervised deep learning
I. Higgins et al.

In the latent factor model

$$\text{Maximize}_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)]$$

the β -VAE adds a constraint or penalty to make the latent representation q_{ϕ} close to a prior, e.g., $p \sim N(0, 1)$, to “disentangle” it.

$$\text{Maximize}_{\phi, \theta} \mathbb{E}_{z \sim q_{\phi}(z|x)} [\log p_{\theta}(x|z)] - \beta \cdot \text{KL}(q_{\phi}(z|x) \| p(z))$$

The original VAE corresponds to $\beta = 1$.

CorrGAN: sampling realistic financial correlation matrices using generative adversarial networks
G. Marti (2019)

GAN to generate *reordered* (with hierarchical clustering) correlation matrices.

Estimation of theory-implied correlation matrices
M. López de Prado (2019)

A hierarchical classification (e.g., GICS) can improve the estimation of the correlation matrix:

- Compute the sample correlation matrix;
- Represent the hierarchy as a tree and use the correlations to define distances between nodes, using $d = \sqrt{\frac{1}{2}(1 - \rho)}$;
- Compute the distances between all pairs of leaves, and convert them back to “correlations”, using $\rho = 1 - 2d^2$;
- Clean the resulting matrix (it may not even be positive definite) using random matrix theory (RMT).

DeepTraX: embedding graphs of financial transactions
C.B. Bruss et al. (2019)

Project the credit card (bipartite) transaction graph to only keep merchants and compute a random-walk based vector embedding (DeepWalk, Node2vec); use as additional features in fraud detection systems.

Testing for multiple bubbles 1: historical episodes of exuberance and collapse in the S&P 500
P.C.B. Phillips et al. (2013)

The SADF test,

$$\text{SADF} = \sup_{0 < t_0 \leq t \leq 1} \text{ADF}_{[t_0, 1]},$$

to detect bubbles (explosive behaviour: bubble-less log-prices should be $I(1)$, not more) can be generalized to account for multiple bubbles by using a moving window (reset after each bubble) instead of an expanding one.

Decomposing value
J. Gerakos and J.T. Linnainmaa (2016)

Decompose the B/P into a component explained by past (5-year) size changes, and a residual – only the former leads to excess returns.

Time-dependent Black-Litterman
M. van der Schans and H. Stehouwer (2016)

Black-Litterman and Kalman filter can be combined: both are (linear, Gaussian) Bayesian updates.

***Listening to chaotic whispers:
a deep learning framework
for news-oriented stock trend prediction***
Z. Hu et al (2017)

Attention-based RNN, not for sequences of words, but for sequences of news (bags of (vector embeddings of words), to forecast trends.

Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises
L. Calès et al. (2018)

The *cross-sectional score* of a portfolio is the probability that a random (uniform in Δ^d) portfolio has worse returns (or is worse for some other portfolio metric). It can be computed with a quasi Monte Carlo method. [This is a special case of P. Burns's random portfolios, which also account for portfolio constraints; the uniform distribution does not give insightful results if there are too many assets.]

Notes on Fano ratio and portfolio optimization
Z. Kakushadze and W. Yu

Daily and monthly Sharpe ratios are not comparable (because of the square root of time rule), but the mean-to-variance ratios (Fano ratios) are comparable across investment horizons. Fano-optimal portfolios are more diversified.

Nonlinearity in stock networks
D. Hartman and J. Hlinka (2018)

Differences between stock networks computed from correlation matrices and mutual information (which requires joint density estimation) may not come from nonlinearities but from non-stationarity and non-Gaussian margins.

Conic CPPI
I. Marquet and W. Schoutens (2017)

A 1-asset investment strategy can be seen as a reinforcement learning problem in which the state is described by the VIX and the current wealth. Instead of maximizing the expected future reward, for the real or the risk-free measure, one can maximize

$$\inf_{Q \in \mathcal{M}} E^Q[\text{future rewards}]$$

which can be computed as the expectation under a concave *distortion function*

$$\psi : [0, 1] \rightarrow [0, 1], \quad \psi(0) = 0, \quad \psi(1)$$

(*Choquet expectation*, expectation under a *non-additive probability*, to emphasize tail loss events and de-emphasize tail gain events)

$$E^\psi[Y] = - \int_{-\infty}^0 \psi(F_Y(y)) dy + \int_0^\infty (1 - \psi(F_Y(y))) dy.$$

The *min-max-var* family of distortion functions is

$$\psi(u) = 1 - (1 - u^{1/(1+\lambda)})^{1+\lambda}.$$

***Equity portfolio risk (volatility) estimation
using market information and sentiment***
L. Mitra et al. (2008)

Implied volatility and change in news sentiment (Ravenpack) have a predictive power in future volatility, and can be used as part of a factor risk model.

Indirect influences in international trade
R. Díaz and L. Gómez

To compute the influence of a country on the trade network, use

- PageRank
- The incidence matrix D (direct influence);
- MicMac: D^k ;
- The heat kernel: $\exp \lambda(D - I)$;
- PWP: $\expm1(\lambda D) / \expm1(\lambda)$

with the trade network

$$D_{A \rightarrow A} = \frac{\text{Exports}_{A \rightarrow B} + \text{Imports}_{B \rightarrow A}}{\text{Exports}_A + \text{Imports}_A}$$

or the offer network

$$D_{A \rightarrow A} = \frac{\text{Exports}_{A \rightarrow B} + \text{Imports}_{B \rightarrow A}}{\text{GDP}_A + \text{Imports}_A}.$$

Dissecting characteristics nonparametrically
J. Freyberger et al. (2017)

Spline regression, with a group lasso penalty, performs better than a linear model (with a lasso penalty) on uniformized variables. [I use GAM boosting to the same effect.]

Taming the factor zoo
G. Feng et al. (2017)

To check if a new factor adds value, to forecast returns, to an already large set of factors $(f_j)_j$, use the *double-selection lasso*:

- Select a small set of factors using the lasso, $(g_i)_i$;
- For each selected factor g_i , select factors f whose covariance with the returns r is correlated with $\text{Cov}(g_i, r)$, using the lasso: $\text{Cov}(g_i, r) \sim \text{Cov}(f, r)$;
- Use all those factors to test the new candidate.

The cross-section of risk and return
K. Daniel et al. (2018)

Separate the priced from the unpriced components of the Fama-French portfolios – removing the industry is sufficient.

***A review of portfolio choice
based on stochastic dominance***
T. Post (2017)

The integrated cdf, used to define stochastic dominance, is the expected shortfall (seen as a function of the threshold): a portfolio λ dominates a portfolio τ if its expected shortfall is better, for all thresholds – equivalently, if it is better for all (increasing, concave) utility functions.

***The dynamic factor model
with an application to global credit risk***
F. Bräuning and S.J. Koopman (2016)

Model the financial network (Granger causality tests for CDSs of US and European banks) as

$$\begin{aligned} f_t &: \text{factors (global, not observed)} \\ z_{ij} &: \text{factor loadings (constant)} \\ \theta_{ijt} &= z'_{ij} f_t \\ \phi_{ijt} &= \text{logistic}(\theta_{ijt}) \\ y_{ijt} &\sim \text{Bernoulli}(\phi_{ijt}) \text{ incidence matrices} \\ \gamma_i &\sim N(\mu, \Sigma) \\ \pi_i &\propto \exp(\gamma_i) \quad k\text{-dimensional} \\ \delta_{ij} &= \pi_i \otimes \pi_j \quad k^2\text{-dimensional} \\ z_{ij} &\sim \text{Multinomial}(\delta_{ij}) \\ f_{t+1} &= \Phi f_t + \xi_t \\ \xi_t &\sim N(0, \Sigma) \\ f_1 &\sim N(0, V) \\ V &= \Phi V \Phi' + \Sigma \end{aligned}$$

***Using dynamic model averaging in state space
representation with dynamic Occam's window
and applications to the stock and gold market***
M. Risse and L. Ohl (2016)

A time-varying regression can be modeled with a state space model and estimated with a Kalman filter; one can make some empirical changes to the variance matrix updates to simplify the computations (scalar variance matrix) or account for stochastic volatility.

Dynamic model averaging (DMA) estimates those models for subsets of the variables, whose forecasts can then be averaged (as in Bayesian model averaging, BMA), with weights depending on the previous weights (shrunk towards uniform weights) and the predictive density of each model.

The *dynamic Occam window* (DOW) does not consider all 2^K models but only a subset, updated at each time step, obtained by adding/removing variables to/from the current model and by keeping only the best ones (as with particle filters).

***Multiperiod portfolio selection
and Bayesian dynamic models***
P. Kolm and G. Ritter (2014)

Assume the transaction-cost-aware portfolio x_t and the ideal portfolio (TC=0) y_t are the hidden and observed states of a hidden Markov chain (HMM) with transition and emission probabilities

$$\begin{aligned} -\log p(x_t | x_{t-1}) &= c_t(x_{t-1}, x_t) \\ -\log p(y_t | x_t) &= \frac{1}{2} \lambda (y_t - x_t)' \Sigma (y_t - x_t). \end{aligned}$$

Treat one asset at a time, in a Gibbs fashion, until convergence (blockwise coordinate descent).

For time, either treat the periods one at a time, or use a particle filter.

Volatility trading
E. Sinclair (2013)

2. There are many measures of volatility (standard deviation of the Brownian motion (with drift and jumps) modeling log-prices): standard deviation of the log-returns (the sample variance gives a biased estimator of volatility: $E[\sqrt{s^2}] < \sqrt{E[s^2]}$, but it can be corrected if we assume Gaussian log-returns), from the absolute returns $\sigma = \sqrt{\frac{\pi}{2}} E[|r|]$, from OHLC data (they can account for drift and jumps, but the true high and low are not observed: these estimators under-estimate volatility), or from high-frequency data (by looking at the time τ needed to achieve a fixed change in price Δ – it is biased, but the bias can be corrected).

There is microstructure noise if the frequency is too high.

Volatility varies through the day.

3. Stylized facts include: volatility clustering, mean reversion ($\sigma_{\text{daily}} > \sigma_{\text{weekly}}$), leverage ($\text{Cor}(\sigma, r) < 0$, with an even lower value if $r < 0$), $\text{Cor}(\sigma, \text{volume}) > 0$, $\sigma \sim \text{LogGaussian}$.

4. Volatility can be forecasted with a GARCH(1,1) model, or its many variants. It can also be compared with the *volatility cone*, the quantiles of the realized volatility as a function of the window size. Overlapping windows give biased estimators, but the bias can be corrected (Hodges-Tompkins).

Fundamental data has some predictive power on volatility: R&D, cash flow volatility, accruals, size, ROA, leverage.

Implied volatility is usually above the volatility forecasts (by 30%): this *variance premium* can sometimes be explained by higher moments.

$$\left(\frac{\text{IV}}{\text{RV}} \right)^2 \approx 1 - \gamma \sigma \text{skew} + \frac{1}{2} \gamma^2 \sigma^2 (\kappa - 3), \quad \gamma = \text{risk aversion}$$

5. PCA suggests that 60% to 80% of the changes in the volatility surface are due to parallel shifts: it suffices to look at ATM IV (but it is tricky to compute: there

is no ATM strike, there is no exact 1-month maturity, there are calls and puts, there are bid and ask prices – interpolate or use the VIX).

volatility surface : $IV \sim \text{strike} + \text{expiry}$
 volatility smile (skew) : $IV \sim \text{strike} |, \text{expiry}$
 volatility term structure : $\text{ATM } IV \sim \text{expiry}$

The VIX is mean reverting (if it were tradable, we could make a profit).

Implied volatility rises before earnings announcements and falls afterwards: buy a front-month straddle 10 days before, sell 1 day before; this is more profitable if analyst dispersion is high (but also look at other firm characteristics). More generally, regardless of the underlying, volatility rises before (planned) news. The expected jump size can be estimated by comparing the IV of the front and second months,

The movement of the IV smile is often described heuristically, as a *sticky strike* (fixed skew: the IV, as a function of the strike, does not change) or a *sticky delta* (floating skew, swimming skew: the IV, as a function of delta, does not change). Each gives rise to an arbitrage: delta-neutral call spread, and delta-neutral OTM call minus put.

The smile, which can be rescaled as $IV/IV_{\text{ATM}} \sim \Delta$, can be explained by differences in supply and demand, presence or absence of hedging (market makers vs other actors), correlation (for indices), skewness, Kurtosis.

The Corrado-Su formula extends the Black-Scholes model to include skewness and kurtosis and explain the smile; it can be used to compute implied skewness and kurtosis.

Back-month IV often over-reacts (Stein effect).

6. Continuous hedging, to ensure $\Delta = 0$, is too costly (the transaction costs are non-zero) and impossible (because of the bid-ask spread). Heuristics include: hedging at regular intervals, hedging to a delta band, hedging when the spot changes.

The trade-off between cost and risk can be measured with a utility function, e.g., $U(w) = -e^{-\gamma w}$, which has a constant risk aversion $\gamma = U''/U'$.

The *Hodges-Neuberger* model does not price the option, but the replication strategy, and includes transaction costs; it only hedges so that the trader be indifferent between the risk of the mis-hedged position and the hedging cost – it re-hedges when Δ leaves a band around/near the Black-Scholes delta. There is no closed-form solution, but an asymptotic, incorrect (centered, symmetric) formula (Whalley-Wilmott) if the transaction costs are small. Zakamouline provides a heuristic (fitted) approximation of the HN bands.

Market impact is

$$F(n) = \text{volatility} \times \sqrt{\text{proportion of the ADV}}.$$

To reduce costs, try to hedge a position on a stock with an index, using β , after aggregating all positions.

To account for jumps, consider static hedging (with other options) rather than dynamic hedging,

7. The P&L is path-dependent.

$$\text{P\&L} = \frac{1}{2}(IV^2 - RV^2) \int e^{-rt} S_t^2 \Gamma dt$$

We do not know the volatility, but prefer it to be biased upward or downward, depending on the market and Γ .

	$\Gamma < 0$	$\Gamma > 0$
trend	low	high
no trend	high	low

We do not hedge continuously.

$$E[\text{P\&L}] = \text{Vega} \times (\sigma_{\text{implied}} - \sigma_{\text{realized}})$$

$$\text{Sd}[\text{P\&L}] \approx \sqrt{\frac{\pi}{4}} \text{Vega} \frac{\sigma}{\text{number of hedges}}$$

8. Trade sizing is often done heuristically (fixed trade size, fixed fraction size).

The **Kelly criterion** can help: take successive iid bets on a fraction of your wealth, win (resp. lose) with returns w (resp. ℓ) with probability p ; after n wins and m losses, your wealth is $w_0(1+fw)^n(1-f\ell)^m$; maximize the expected utility, assuming logarithmic utility:

$$f = \frac{pw + (1-p)\ell}{w\ell}.$$

This can be generalized to continuous outcomes: $f \approx r/\sigma^2$.

While optimal in the long term, trading at the Kelly rate is a very volatile strategy, which can take centuries or millennia to beat other strategies – many suggest trading a fraction of the Kelly rate (never more: both returns and volatility would be worse).

There are many progressive betting systems: positive (increase your bet after you win, e.g., Kelly), negative (increase your bet after you lose) or mixed (Oscar: bet one unit; if you lose, bet the same amount; when you win, increase the amount by 1; stop when you have won one unit).

Browne dynamic sizing maximizes the probability of reaching a target wealth within a specified time (Kelly sizing is constant).

Optimal sizing for an OU process is $-w\sigma/2$, where w is the wealth and σ the distance (in standard deviations) to the long-term mean.

9. The variance of the Sharpe ratio is high.

$$\text{Var}[\text{IR}] = \frac{1}{\sqrt{T}}(1 + \frac{1}{2}\text{IR}^2)$$

Also consider other performance ratios: Sortino, Omega.

Measure the persistence of the performance, e.g., by comparing different periods, or with the Hurst exponent.

11. To profit from the variance premium ($IV > RV$), short *index* volatility, by selling 20- Δ straddles or strangles on QQQ, second month, Δ -hedged, rebalanced monthly, when $VIX < 35$.

To profit from the correlation premium (short correlation, dispersion trading): sell index vol, buy component vol.

To profit from the skewness premium, sell 30- Δ risk reversals (sell the put, buy the call, Δ -hedge), or sell skew swaps (the payoff is the difference between realized and implied skew).

12. A model-free implied variance can be computed from all the calls and puts,

$$\text{Var}_{[T_1, T_2]} = \frac{2}{T_2 - T_1} \int_0^\infty \frac{C_{T_2}(Xe^{rT_2}) - C_{T_1}(Xe^{rT_1})}{X^2} dX$$

$$\text{Var}_{[0, T]} = \frac{2}{T} e^{rT} \left[\int_0^F \frac{P_T(X)}{X^2} dX + \int_F^\infty \frac{C_T(X)}{X^2} dX \right]$$

where F is the forward price.

The (cash) VIX is a finite sum approximation of those integrals; it is not tradable, but there are VIX futures; the cash VIX is a good predictor of VIX futures.

There are short- and long-term volatility ETNs (VXX, VXZ). The implied volatility term structure is

$$\text{IVTS} = \frac{\text{VIX}}{\text{VXV}},$$

where VIX and VXV are estimating the 1- and 3-month IV. A calendar strategy uses the IVTS to combine VXX and VXZ.

A calendar put spread, in contango, buys the front month ATM put and sells the second month put with the same strike.

13. *Leveraged ETFs* are computed using daily returns, compounded (if you expect to get λ times the monthly performance, you will be disappointed): they exhibit *volatility drag* (a rise of 10% and a drop of 10% and the effect is amplified as the leverage increases. To use it, buy one dollar of L and sell λ dollars of S .

$$L = L_0 \left(\frac{S}{S_0} \right)^\lambda \exp \left[-\frac{1}{2} \lambda (\lambda - 1) \sigma^2 t \right]$$

Deep analytics

B. Huge and A. Savine (2019)

Pricing options often requires lengthy Monte Carlo simulations: this is not scalable. Replace those computations with a trained deep learning model. The Longstaff-Schwartz model is an early example, but uses a linear model. Use *differential regularization*: two loss terms, one for the value of the option, the other for its gradient (sensitivities, Greeks); this has the added advantage of improving extrapolation. With code examples.

Modern Computational Finance: AAD and Parallel Simulations with professional implementation in C++

A. Savine (2019)

C++ book on parallel processing, with applications to automatic differentiation (with xVA computations in mind).

Edgeworth trading on networks

D. Cassese and P. Pin

Pure exchange model (no production and consumption), with n goods, in which each agent has a utility function determining how he trades.

Kernels and regularization on graphs

A.J. Smola and R. Kondor (2003)

The **graph Laplacian** $L = D - W$ is defined by

$$f' L f = \frac{1}{2} \sum_{i-j} (f_i - f_j)^2 \quad \text{for } f \in \mathbf{R}^n.$$

The pseudo-norm $\|f\|_L = \langle f, Lf \rangle$ is a discrete analogue of

$$\|f\|_\Delta = \langle f, \Delta f \rangle = \langle \nabla f, \nabla f \rangle = \int \|\nabla f\|^2.$$

Up to row/column/sum operations, the graph Laplacian (or the **normalized graph Laplacian**, $\tilde{L} = I - D^{-1/2} W D^{-1/2}$, whose spectrum is in $[0, 2]$) is the only permutation-invariant linear operator.

By analogy with radial basis function regularization

$$\Omega(f) = \langle f, r(\Delta) f \rangle = \int |\hat{f}(\omega)| r(\|\omega\|^2) d\omega$$

$$k(x, y) = \langle k(x, \cdot), r(\Delta) k(y, \cdot) \rangle$$

one can define regularization for functions on a graph, $f : V \rightarrow \mathbf{R}$, i.e., $f \in \mathbf{R}^{|V|}$,

$$\Omega(f) = \langle f, r(\tilde{L}) f \rangle \quad \text{regularizer}$$

$$K = r(\tilde{L})^{-1} \quad \text{kernel.}$$

For instance,

$r(\lambda) = 1 + \sigma^2 \lambda$	regularized Laplacian
$r(\lambda) = \exp(\sigma^2 / 2 \lambda)$	diffusion
$r(\lambda) = (a - \lambda)^{-1}$	1-step random walk, $a \geq 2$
$r(\lambda) = (a - \lambda)^{-p}$	p -step random walk
$r(\lambda) = \cos(\lambda \pi / 4)^{-1}$	inverse cosine.

To compute Ω or K , use a low-degree polynomial approximation of $r(\lambda)$ or $r(\lambda)^{-1}$: Taylor expansion around zero, $\exp(B) \approx (I + B/n)^n$, Chebychev polynomials on $[0, 2]$, etc. (more simplifications are available for product graphs).

The Eikonal equation

$$\begin{aligned} |\nabla\phi| &= 1 \\ \phi|_\gamma &= 0 \end{aligned}$$

computes the distance to a subset $\gamma \subset M$ of a Riemannian manifold but, being nonlinear and hyperbolic, it is difficult to solve (*fast marching algorithm*).

Varadhan's formula gives the distance from the heat kernel,

$$\phi(x, y) = \lim_{t \rightarrow 0} \sqrt{-4t \log k_{t,x}(y)}$$

(interpreting the heat equation with random walks: after a small time t , a walk reaching y from x has not had time to deviate much from the geodesics), but it is very sensitive to errors in the kernel (especially if t is small).

The heat method

- Integrates the heat flow $\dot{u} = \Delta u$ for some fixed time t ;
- Evaluates the vector field $X = -\nabla u / |\nabla u|$ – we only keep the direction: the amplitude is imprecise and, for geodesics, is known to be one;
- Find the corresponding distance by solving $\nabla\phi = X$, *i.e.*, $\text{Argmin}_\phi \int |\nabla\phi - X|$, *i.e.*, $\Delta\phi = \nabla \cdot X$ (Poisson).

The emergence of spectral universality in deep networks

F. Pennington et al.

To study the spectrum of the Jacobian *doutput/dinput* of a neural net (if all the singular values are close to 1, learning is faster), use **free probability theory**.

$$\rho_X(\lambda) = \left\langle \frac{1}{N} \sum_{i=1}^n \delta(\lambda - \lambda_i) \right\rangle_X \quad \text{Limiting spectral density}$$

$$G_X(z) = \int_{\mathbf{R}} \frac{\rho_X(t)}{z - t} dt \quad \text{Stieltjes transform } (z \notin \mathbf{R})$$

$$M_X(z) = zG_X(z) - 1 \quad \text{Moment generating function}$$

$$M_X^{-1} \quad \text{Its inverse}$$

$$s_X(z) = \frac{1 + z}{zM_X^{-1}(z)} \quad \text{S-transform}$$

$$S_{AB}(z) = s_A(z)s_B(z)$$

In the other direction, use

$$\rho_X(\lambda) = -\frac{1}{\pi} \lim_{\varepsilon \rightarrow 0^+} \text{Im } G_X(\lambda + i\varepsilon).$$

Spherical CNNs

T.S. Cohen (2018)

Convolutional layers are translation-invariants; they could be computed with a fast Fourier transform (FFT) – in practice, they are not, because the FFT is slow for

small filters. The Fourier transform can be generalized to signals on the sphere \mathbf{S}^2 (drone vision, molecular structure, weather modeling) and rotation-invariant transformation, but since \mathbf{S}^2 is not a group, the Fourier transform yields a signal on $\text{SO}(3)$ (in further layers, we stay in $\text{SO}(3)$: it is already a group.)

For signals f, g on the sphere, one can define a “correlation” $f * g$.

$$x \in \mathbf{S}^2$$

$$f, g : \mathbf{S}^2 \rightarrow \mathbf{R}^k$$

$$Q, R \in \text{SO}(3)$$

$$\langle f, g \rangle = \int_{\mathbf{S}^2} \langle f(x), g(x) \rangle_{\mathbf{R}^k} dx$$

$$(L_R f)(x) = f(R^{-1}x) \quad \text{action of } \text{SO}(3) \text{ on } \mathbf{S}^2$$

$$(f * g)(R) = \langle L_R f, g \rangle = \langle f, L_R g \rangle \quad \text{correlation}$$

Those definitions extend to functions on $\text{SO}(3)$.

$$f, g : \text{SO}(3) \rightarrow \mathbf{R}^k$$

$$\langle f, g \rangle = \int_{\text{SO}(3)} \langle f(x), g(x) \rangle_{\mathbf{R}^k} dx$$

$$(L_R f)(Q) = f(R^{-1}Q)$$

$$(f * g)(R) = \langle L_R f, g \rangle$$

Spherical convolution can be computed with the generalized FFT.

L4: practical loss-based stepsize adaptation for deep learning

M. Rolínek and G. Martius (2018)

Choose the step size to have the same forecasted loss improvement at each step. If the loss were linear, we could go to the minimum in one step:

$$\begin{aligned} \text{loss}(\theta_0 + \eta v) &= \text{loss}(\theta_0) + \eta g' \cdot v = L_{\min} \\ \eta &= -\frac{\text{loss}(\theta_0) - L_{\min}}{g' \cdot b}. \end{aligned}$$

In practice, the step can be $\alpha \eta v$, where g is the gradient, v the direction (gradient, momentum, etc.), $\alpha = 0.15$ and where L_{\min} is some fraction (say, 0.9) of the lowest loss seen so far.

Stochastic hyperparameter optimization through hypernetworks

J. Lorraine and D. Duvenaud

Learn the function $w^*(\lambda) = \text{Argmin}_w \text{loss}(w, \lambda)$, where λ are the hyperparameters and w the parameters.

Reordering rows for better compression: beyond the lexicographic order

D. Lemire et al.

Lexicographic order does not give the optimal RLE compression: pick random elements close in Hamming distance (*i.e.*, get close to a Gray code).

***Geometric properties of local dynamics
in Hamiltonian systems: the generalized
alignment index (GALI) method***
C. Skokos et al. (2018)

The generalized alignment index (GALI), in a Hamiltonian system, is the volume of a parallelepiped of k initially orthogonal unit vectors, whose magnitude is normalized to 1 at each time step; for chaotic orbits, it tends exponentially fast to zero, with exponents related to Lyapunov exponents.

***Constant-step-size least-mean-square
bias-variance trade-offs
and optimal sampling distributions***
A. Défossez and F. Bach (2014)

Average (Cesàro) constant-step-size stochastic gradient descent.

***A practical tutorial on autoencoders
for nonlinear feature fusion:
taxonomy, models, software and guidelines***
C. Charte et al. (2018)

A **contractive autoencoder** has a penalty on the Frobenius norm of the Jacobian of the encoder.

Corentropy (minus a Gaussian kernel \mathbf{V}), as a loss function,

$$\text{loss}(u, v) = - \sum_k \phi\left(\frac{u_k - v_k}{\sigma}\right)$$

gives robust autoencoders.

Feature fusion is a synonym for dimension reduction (PCA, LDA, kernel PCA, MDS, Isomap, Laplacian eigenmaps, RBM)

***Analysis of multivariate time-series
using the MARSS package***
E.E. Holmes et al. (2018)

hidden $x_t = Bx_{t-1} + u + Cc_t + w_t \quad w_t \sim N(0, Q)$
observed $y_t = Zx_t + a + Dd_t + v_t \quad v_t \sim N(0, R)$

***Matrix profile I:
all pairs similarity joins for time series:
a unifying view that includes motifs,
discords and shapelets***
C.C.M. Yeh et al. (2016)

The **matrix profile** of a time series is the time series of distances to the nearest neighbour.

$$\text{mp}_t = \underset{|t-s|>a}{\text{Min}} \, d(x_{t:t+n}, x_{s:s+n})$$

It summarizes the information in the distance matrix $(d(x_t, x_s))_{t,s}$.

To be robust to outliers, take the k th nearest neighbour.

To convert DNA to a numeric sequence, add 2, 1, -1 , -2 for A, G, C, T.

Use weights to account for prior information (regions more or less likely to contain interesting motifs).

$$\text{cmp}_i = \text{mp}_i + (1 - \text{av}_i) \text{Max}(\text{mp})$$

Applications include:

- Motif detection: low values;
- **Discords** (outliers): high values;
- Segmentation, by looking at where the nearest neighbours are and finding the point with the fewest crossings.

The matrix profile can be computed efficiently ($n \log n$, thanks to the FFT, if we use the z -normalized Euclidean distance, *i.e.*, the correlation).

Implementations in Python (**stumpy**, etc.) and R (**tsmp**); 14 more papers in the series.

***A complexity-invariant distance measure
for time series***
G.E. Batista et al.

Pairs of complex objects tend to be more distant, for most distance measures, than pairs of simple objects. There are already distance measures invariant to offset, scale (correlation), warping (DTW), phase, occlusion (missing subsequence):

$$\text{CID}(f, g) = \|f - g\|_2 \frac{\text{Max}\{\|f'\|_2, \|g'\|_2\}}{\text{Min}\{\|f'\|_2, \|g'\|_2\}}$$

is invariant to complexity changes, if the complexity of f is measured as $\|f'\|_2$.

***A brief introduction to machine learning
for engineers***
O. Simeone (2017)

1. There are several approached to statistics:

- Frequentist (no prior; point estimate);
- Bayesian (prior; posterior distribution);
- MAP (prior distribution, *i.e.*, regularization; point estimate);
- MDL (penalty for complexity; point estimate).

2. **No free lunch theorem.** One cannot learn rules that generalize to unseen examples without making assumptions (*inductive bias*) about the data generation mechanism.

Supervised learning looks for a predictive algorithm minimizing generalization loss.

Discriminative models directly model $p(y|x)$.

Generative models model $p(x, y)$ (sometimes by separately modeling $p(y)$ and $p(x|y)$), from which we can compute $p(y|x)$ – there is a higher risk of incorrect assumptions, but it is easier to deal with missing values.

The Bayesian approach allows model selection without validation, with the **marginal likelihood**.

f_α	prior
$\theta \sim f_\alpha$	parameters
g_θ	model likelihood
$x \sim g_\theta$	data
$g(x) := E_\theta [g_\theta(x)]$	marginal likelihood

Empirical Bayes estimates the parameter prior from the data.

The **Kullback-Leibler divergence** is the expectation of the log-likelihood-ratio between two distributions, $\log p(x)/q(x)$, wrt the distribution in the numerator. It is non-negative; it is zero iff $p = q$. The log-likelihood is $\text{KL}(\text{data} \parallel \text{model})$.

$$\begin{aligned} \text{KL}(p \parallel q) &= \mathbb{E}_{x \sim p} \left[\log \frac{p(x)}{q(x)} \right] \\ H(p \parallel q) &= \mathbb{E}_{x \sim p} [-\log q(x)] && \text{cross-entropy} \\ H(p) &= \mathbb{E}_{x \sim p} [-\log p(x)] && \text{entropy} \end{aligned}$$

3. Exponential family (or **log-linear**) models are of the form

$$\begin{aligned} p(x|\eta) &\propto \exp(\eta' u(x)) m(x) \\ \log p(x|\eta) &= \eta' u(x) + \log m(x) - A(\eta) \end{aligned}$$

where

η	natural parameters
u	sufficient statistics
m	base measure
$\mu = \mathbb{E}_{x \sim p(\cdot \eta)} [u(x)]$	mean parameters.

The Gaussian model $N(\mu, \sigma^2)$ is exponential, with

$$u(x) = \begin{pmatrix} x \\ x^2 \end{pmatrix} \quad \eta = \begin{pmatrix} \mu/\sigma^2 \\ -1/2\sigma^2 \end{pmatrix} \quad m = 1.$$

The natural and mean parameters are related through the gradient of the partition function.

$$\begin{aligned} \nabla_\eta A(\eta) &= \mu \\ \nabla_\eta^2 A(\eta) &= \text{Cov } u(x) = I_\eta \end{aligned}$$

The exponential family $p(x|\eta)$ is the **maximum entropy** distribution under the constraints $E[u(x)] = \mu$ (and the natural parameters are the corresponding Lagrange multipliers).

The gradient of the log-likelihood is $\nabla_\eta \ell = \hat{\mu} - \mu$.

In Bayesian models, one often chooses *conjugate priors*, i.e., priors such that priors and posteriors belong to the same family (Beta-Bernoulli, Dirichlet-Categorical, Gaussian-Gaussian, etc.).

One can vary the prior, to see, for instance, how strong it should be to change the MAP estimator.

Energy-based models are of the form

$$p(x|\eta) \propto \exp - \sum_c E_x(x_c|\eta);$$

they generalize the exponential families (for which E_c is linear). For instance, $E(x|\eta) = \log(1 + (\eta'x)^2)$ gives a Student T distribution; Markov models are another example.

A **generalized linear model** (GLM) is a model from an exponential family whose parameter η depends linearly on the predictors, $\eta = Wx$ or $\eta = W\phi(x)$.

4. Classification

Discriminative deterministic (classification) models (SVM, k -NN) output binary results.

Discriminative probabilistic models (GLM) output probabilities $P[\text{label}|\text{feature}]$.

Generative probabilistic models model the joint probability $P(\text{label}, \text{feature})$, often separating $P(\text{label})$ and $P[\text{features}|\text{label}]$; examples include QDA

$$\begin{aligned} t &\sim \text{Bernoulli}(\pi) \\ x|t = k &\sim N(\mu_k, \Sigma_k) \end{aligned}$$

or LDA (when Σ_k does not depend on k).

5. Statistical learning theory. We want to minimize the *generalization error*

$$\mathbb{E}_{(x,y) \sim p} \text{loss}(t, \hat{t}(x))$$

but the data distribution p is not known. Instead, we can use the sample distribution and minimize the *empirical risk*

$$\mathbb{E}_{(x,y) \sim \text{Data}} \text{loss}(y, \hat{t}(x)).$$

A learning rule is **probably approximately correct** (PAC) if

$$\forall p \in \mathcal{H} \quad \mathbb{P}_{\substack{\text{Data} \sim p \\ \text{size}(\text{Data}) \geq N}} [\text{loss}_p(\hat{t}) \leq \text{loss}_p(t^*) + \varepsilon] \geq 1 - \alpha$$

where N , the *sample complexity*, only depends on ε and α (not p).

Finite hypothesis classes \mathcal{H} are **PAC-learnable** via empirical risk minimization (ERM) with sample complexity

$$\begin{aligned} N &= \frac{2 \log |\mathcal{H}| + 2 \log(2/\delta)}{\varepsilon^2} \\ N &= \frac{\log |\mathcal{H}| + \log(1/\delta)}{\varepsilon} (?) \\ N &= \Theta \left(\frac{\dim_{\text{VC}} \mathcal{H} + \log(1/\delta)}{\varepsilon^2} \right). \end{aligned}$$

Dimension- d (quantized) models have sample complexity proportional to $d + 1$.

The *minimax redundancy* is another measure of model capacity.

$$\Delta R(\mathcal{H}) = \min_{q \in \mathcal{H}} \max_{p \in \mathcal{H}} \text{KL}(p \parallel q)$$

Structural risk minimization combines model selection and hypothesis testing by minimizing an upper bound on the generalization loss

$$\text{loss}_p(\hat{t}) \leq \text{loss}_{\text{Data}}(\hat{t}) + \sqrt{\frac{\log 2 |\mathcal{H}| / \delta}{2N}}.$$

6. The EM algorithm

x : observed
 z : latent
 θ : parameters

alternates between two steps

- M step: point estimate of $\theta|x, z$;
- E step: distribution estimate of $z|x, \theta$;

It can also be motivated using the **evidence lower bound**(ELBO)

$$\begin{aligned} \log p(x|\theta) &= \log \sum_z p(x, z|\theta) \\ &= \log \sum_z q(z) \frac{p(x, z|\theta)}{q(z)} && \text{Importance sampling} \\ &> \sum_z q(z) \log \frac{p(x, z|\theta)}{q(z)} && \text{Jensen} \\ &= \mathcal{L}(q, \theta) \end{aligned}$$

- E step: $q \leftarrow \text{Argmax}_q \mathcal{L}(q, \theta)$ (with θ fixed), *i.e.*, $q(z) = p(z|x, \theta)$;
- M step: $\theta \leftarrow \text{Argmax}_\theta \mathcal{L}(q, \theta)$.

Probabilistic PCA (PPCA) can be estimated with the EM algorithm.

$$\begin{aligned} z &\sim N(0, I) \\ x|z &\sim N(Wx + \mu, \sigma^2 I) \end{aligned}$$

In unsupervised learning, the loss is often the divergence between the model and the data. The Kullback-Leibler divergence is not always the best choice: minimizing $\text{KL}(\text{data}||\text{model})$ tends to produce blurry estimates.

Generative adversarial networks (GAN) generalize generative directed models: they learn at the same time a divergence and a generative model,

$$\text{Min}_\theta \text{Max}_T \text{E}_{x \sim p} [T(x)] - \text{E}_{x \sim q_\theta} [g(T(x))]$$

with T a neural net and g fixed

$$\begin{aligned} g(t) &= -\log(1 - e^t) && \text{GAN} \\ g(t) &= 1 - t && \text{WGAN.} \end{aligned}$$

Helmoltz machines are multilayer extensions of generative directed models.

7. Probabilistic graphical models, directed (Bayesian networks) or not (Markov random fields, MRF) encode conditional independence relations.

8. Variational inference (VI) approximates the posterior as an I-projection.

$$\begin{aligned} p(z|x) &= \underset{q(z)}{\text{Argmin}} \text{KL}(q(z) || p(z|x)) \\ &= \underset{q(z)}{\text{Argmin}} \text{KL}(q(z) || p(z, x)) + \log p(x) \\ &= \underset{q(z)}{\text{Argmin}} \text{KL}(q(z) || p(z, x)) \\ &= \underset{q(z)}{\text{Argmin}} - \text{E}_{z \sim q} [\log p(z, x)] - H(q) \\ &= \underset{q(z)}{\text{Argmin}} - \mathcal{L}(q) \\ &\approx \underset{q \in \mathcal{F}}{\text{Argmin}} - \mathcal{L}(q) \end{aligned}$$

The I-projection is mode-seeking, the M-projection zero-avoiding.

$$\begin{aligned} \text{pr}_I(p) &= \underset{q \in \mathcal{F}}{\text{Argmin}} \text{KL}(q||p) \\ \text{pr}_M(p) &= \underset{q \in \mathcal{F}}{\text{Argmin}} \text{KL}(p||q) \end{aligned}$$

The α -divergence

$$D_\alpha(p||q) = \frac{\sum_x \alpha p(x) + (1 - \alpha)q(x) - p(x)^\alpha q(x)^{1-\alpha}}{\alpha(1 - \alpha)}$$

interpolates between $\text{KL}(p||q)$ (for $\alpha = 1$) and $\text{KL}(p||q)$ ($\alpha = 0$).

Most likely transformations: the mlt package
T. Hothorn (2018)

The regression function of $Y|X$ is often defined as $f(x) = E[Y|X = x]$, without any distributional assumption, but it could be written, with a Gaussian model,

$$Y|X = x \sim N(f(x), \sigma^2)$$

or (this immediately allows censored data)

$$P[Y \leq y|X = x] = F\left(\frac{y - f(x)}{\sigma}\right)$$

or, more generally

$$P[Y \leq y|X = x] = F(h(y) - f(x))$$

where $h(y) = \theta' a(y)$, with θ to be estimated and a basis functions.

Adaptive gradient methods
with dynamic bound of learning rate
L. Luo et al. (2019)

Adaptive optimization algorithms (Adam, RMSProp, etc.) can choose extreme learning rates, leading to poor performance – instead, bound the learning rates to a shrinking interval, *i.e.*, progressively move towards plain stochastic gradient descent.

The Bloom clock **L. Ramabaja**

The *vector clock* provides a partial order on the events in a distributed system:

- Each node keeps a vector of size n , with one coordinate for each node;
- When a node sends an event, it increments its coordinate and broadcasts its vector;
- When a node receives a vector, it increments its coordinate and replaces its vector with the elementwise maximum of the two.

The vector clock can be bloomified.

***f*-GAN: training generative neural samplers using variational divergence minimization**

An *f*-GAN looks for a generator g_θ minimizing the *f*-divergence

$$D_f(p_{\text{data}} \| q_\theta) = \mathbb{E}_{x \sim q} \left[f \left(\frac{p(x)}{q(x)} \right) \right].$$

Since

$$\begin{aligned} D_f(p \| q) &\geq \sup_T \mathbb{E}_{x \sim p} [T(x)] - \mathbb{E}_{x \sim q} [f^*(T(x))] \\ &= \mathbb{E}_{x \sim p} [T(x)] - \mathbb{E}_{x \sim q} [f^*(T(x))] \end{aligned}$$

for $T(x) = f'(p(x)/q(x))$, we can instead try the saddle points of

$$F(\omega, \theta) = \mathbb{E}_{x \sim p_{\text{data}}} [T_\omega(x)] - \mathbb{E}_{x \sim q_\theta} [f^*(T_\omega(x))]$$

which can be approximated with samples from p and q .

Holographic embeddings of knowledge graphs **M. Nickel et al.**

Knowledge graphs (RDF triples, SVO triples) can be embedded in a vector space by learning vector representations of subject, object and predicate,

$$\begin{aligned} \text{score}(s, p, o) &= -d(e_s + r_p, e_o) \\ \text{or } P[s, p, o] &= \sigma(r'(e_s \circ e_o)). \end{aligned}$$

The composition operator $\cdot \circ \cdot$ can be one of

$$\begin{aligned} a \circ b &= a \otimes b \\ a \circ b &= \psi(W(a \oplus b)) \quad \begin{array}{l} \text{concatenation, projection,} \\ \text{non-linearity} \end{array} \\ a \circ b &= a \star b = \left(\sum_i a_i b_{i+k \bmod d} \right)_k \end{aligned}$$

(for the tensor product, since $r'(a \otimes b) = a' R b$, one can ask that R be diagonal, but this makes all relations symmetric).

The **circular correlation** \star can be seen as a “compression” of the tensor product.

Distributed convex optimization with many non-linear constraints **J. Giesen and S. Laue**

ADMM

$$\begin{aligned} \text{Find} & \quad x, z \\ \text{To minimize} & \quad f(x) + g(z) \\ \text{Such that} & \quad Ax + Bz = c, \end{aligned}$$

often used (on a distributed infrastructure) to minimize separable functions $f(\theta) = \sum_i f_i(\theta)$,

$$\begin{aligned} \text{Find} & \quad x, z \\ \text{To minimize} & \quad \sum_i f_i(x_i) \\ \text{Such that} & \quad \forall i \ x_i = z \end{aligned}$$

can also handle nonlinear convex *inequalities* (they keep the problem convex).

Link prediction based on graph neural networks **M. Zhang and Y. Chen (2018)**

Link prediction heuristics (common neighbours, Katz index) make strong assumptions on when two nodes are linked; instead, learn a function mapping subgraph patterns to link existence (SEAL).

Discovering and deciphering relationships across disparate data modalities **Y.T. Vogelstein et al.**

To test if $X \perp\!\!\!\perp Y$:

- Compute the distance matrices (between observations) for X and Y ; center them: A, B ;
- Compute the corresponding k -NN graphs: G_k, H_k (binary matrices);
- Compute the local correlations

$$t_{k\ell} = \sum_{ij} A_{ij} B_{ij} G_{kij} H_{\ell ij};$$

- Smooth $(k, \ell) \mapsto t_{k\ell}$ and compute its maximum t^* ;
- Use a permutation test to see if it is significant.

Minimum Rényi entropy portfolios **N. Lassance and F. Vrans (2018)**

Rényi entropy combines variance, skewness and kurtosis (as shown by a Gram-Charlier expansion): it can be used as a risk measure.

DGM: a deep learning algorithm for solving partial differential equations **J. Sirignano and K. Spiliopoulos (2018)**

Deep learning tools can provide mesh-free solutions to PDEs – meshes quickly become unusable as dimension increases.

Statistical inference using the Morse-Smale complex **Y.C. Chen et al. (2015)**

The **Morse-Smale complex** of a function $f : X \rightarrow \mathbf{R}$ decomposes the space X with the intersections of the

ascending and descending manifolds, *i.e.*, the basins of attractions of the gradients of f and $-f$.

Mean-shift clustering (or **mode clustering**) uses the ascending manifolds as clusters.

Morse-Smale regression estimates a linear model in each cell of the Morse-Smale complex of a kernel estimator $f(x) = E[Y|X = x]$.

The **Morse-Smale signature graph** (of a density estimate, a regression function, or a difference of densities) has maxima and minima as nodes, Morse-Smale cells as edges, the slope of a linear approximation as edge weights, and uses multidimensional scaling (MDS) for node coordinates.

The **Morse-Smale energy test** compares two probability distributions, using half the data to estimate the Morse-Smale complex of the difference of densities, and the other half to perform an energy (distance correlation) test in each cell, with a Bonferroni correction.

Morse-Smale regression **S. Gerber et al. (2013)**

The **Morse-Smale complex** of a function $f : X \rightarrow \mathbf{R}$ known on a finite set of points can be computed from the k -nearest neighbour graph, following the paths of steepest descent or ascent (*quick-shift* algorithm).

The cell of a new point can be predicted using a kernel density estimator or a support vector machine (SVM).

Morse-Smale regression fits a linear model in each cell, either with discontinuities

$$\hat{f}(x) = a_i + b_i \quad \text{if } x \in C_i$$

or with weights from the cell prediction model

$$\hat{f}(x) = \sum_i P[x \in C_i](a_i + b_i x).$$

The *persistence* of a critical point x is $\text{Min}_y \|f(x) - f(y)\|$, where the minimum is over the set of critical points y in cells adjacent to x ; this defines a filtration of Morse-Smale complexes (use the BIC to select the persistence level, *i.e.*, the number of cells).

Visual exploration of high-dimensional scalar functions **S. Gerber et al. (2011)**

Dimension reduction can help understand high-dimensional clouds of points. For functions (scalar fields) defined on an intrinsically high-dimensional cloud of points (or models with many variables interacting in complicated ways):

- Build the k -nearest graph of the cloud of points (k should increase with dimension);
- Compute the corresponding Morse-Smale complex; one can fine-tune the level of detail – start at the coarsest level;

- For each Morse-Smale crystal, estimate an (inverse) local (kernel) regression $X \sim y$: it is a path from minimum to maximum;
- For each such path r , compute a few geometric summaries: sensitivity dr/dy , average distance to the curve, sampling density – these form bands around the curves;
- Project those curves onto the first principal component of the extrema.

Data analysis with the Morse-Smale complex: the msr package for R **S. Gerber et al.**

Implementation of Morse-Smale regression and inverse regression.

Link prediction in complex networks: a survey **L. Lü and T. Zhou (2010)**

Predict missing links in an (undirected, homophilic) network using local or global similarity measures:

- The number of common neighbours, $|\Gamma(x) \cap \Gamma(y)|$, divided by 1, $k_x + k_y$, $k_x k_y$, $\sqrt{k_x k_y}$, $\text{Max}\{k_x, k_y\}$, $\text{Min}\{k_x, k_y\}$, $|\Gamma(x) \cup \Gamma(y)|$;
- $k_x k_y$ (preferential attachment – this uses too little information);
- The number of common neighbours, with less weight for the more promiscuous ones,

$$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{k_z}, \quad \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log k_z}$$

- PageRank (random walk with restarts);
- SimRank;
- Katz: $(I - \beta A)^{-1}$;
- The average commute time $(L^\dagger)_{xx} + (L^\dagger)_{yy} - 2(L^\dagger)_{xy}$, $L = D - A$;
- Matrix forest” $(I + \alpha L)^{-1}$;
- Cosine: $(L^\dagger)_{xy} / \sqrt{(L^\dagger)_{xx}(L^\dagger)_{yy}}$.

Statistical models are another (slow, inaccurate, not scalable) option:

- *Hierarchical structure model*: the nodes are the leaves of a dendrogram; each internal node x is assigned a number $p_x \in [0, 1]$, and the link probability is $P[x-y] = P_{x \wedge y}$, where $x \wedge y$ is the lowest common ancestor of the leaves x and y ;
- *Stochastic block model*.

The link prediction problem for social networks

D. Liben-Nowell and J. Kleinberg (2004)

Forecast new links in a (co-authorship network using node similarity (Katz, Adamic-Adar, etc.), computed from the adjacency matrix, or a low-rank approximation, or a cleaned version (pruned of low-similarity edges).

**Link prediction in evolving networks
based on popularity of nodes**
T. Wang et al.

To predict links in (homophilic) evolving networks, use the *popularity* of the nodes (proportion of recent edges): active nodes attract more links.

**Link prediction in complex networks:
a mutual information perspective**
F. Tan et al.

Nodes whose neighbours tend to be connected tend to be connected.

$$\Gamma(x) = \{\text{neighbours of } x\}$$

$$k_x = |\Gamma(x)|$$

$$M = \text{Number of edges}$$

$$N_{\Delta z} = \# \text{ of connected pairs of neighbours of } z$$

$$N_{\Lambda z} = \# \text{ of disconnected pairs of neighbours of } z$$

$$p_{xy} = \frac{1 - \binom{M-k_y}{k_x}}{\binom{M}{k_x}}$$

$$I_{xy} = -p_{xy} \log p_{xy}$$

$$I_z = \text{Mean}_{x,y \in \Gamma(z)} \log \frac{\binom{M}{k_x}}{\binom{M}{k_x} - \binom{M-k_y}{k_x}} + \log \frac{N_{\Delta z}}{N_{\Delta z} + N_{\Lambda z}}$$

$$S(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} I_z - I_{xy}$$

**Supervised random walks: predicting
and recommending links in social networks**
L. Backstrom and J. Leskovec (2011)

Personalized-PageRank-like measure of similarity computed from a weighted graph, where the (learned) edge weights are computed from edge and node features; train with a hinge loss on $P_{i \rightarrow j} < P_{i \rightarrow k}$, where $i \rightarrow k$ is an edge and $i \rightarrow j$ is not.

**DeepWalk:
online learning of social representations**
B. Perozzi et al. (2014)

Apply *word2vec* (skipgram with negative sampling) to random walks on a graph to get a vertex embedding. Use a Huffman-coding-based *hierarchical softmax* to deal with the large number of nodes.

**node2vec:
scalable feature learning for networks**
A. Grover and J. Leskovec (2016)

Random walks on a graph (depth-first search, DFS) provide an alternative to the traditional notion of neighbourhood (breadth-first search, BFS). Use a *biased random walk* with different probabilities for going back, moving to a neighbour common to the previous node, moving to another neighbour.

Structural deep network embedding
D. Wang et al. (2016)

Learn an *autoencoder* $x_i \mapsto y_i \mapsto \hat{x}_i$ where x_i is the i th row of the adjacency matrix, with

$$\text{loss} = \sum_{(ij) \in E} \|y_i - y_j\|^2 + \sum_{(ij) \in E} \|\hat{x}_{ij} - x_{ij}\|^2 + \text{regularizer};$$

the hidden layer y provides a node embedding.

LLE, IsoMap, Laplacian eigenmaps provide other (shallower) embeddings.

**LINE:
large-scale information network embedding**
J. Tang et al.

Find a low-dimensional embedding u_i of the nodes to minimize

$$- \sum_{(ij) \in E} w_{ij} \log \frac{1}{1 + e^{-u_i' u_j}} \quad \text{or} \quad w_{ij} \log \frac{e^{u_i' u_j}}{\sum_k e^{u_i' u_k}}$$

(KL divergence between the probability distributions inside the logarithms and $\hat{p}_{ij} = w_{ij} / \sum_{k \ell} w_{k \ell}$ or $\hat{p}_{ij} = w_{ij} / \sum_k w_{ik \cdot}$)

**Link prediction via subgraph embedding-based
convex matrix completion**
Z. Cao et al. (2018)

Compute node representations using word2vec (on random walks) or GloVe (on PPMI = $(A + A^2)/2$) on the graph whose nodes are egonets (of radius r) and with edges between egonets in the radial context (of radius R) of a node.

**GraRep: learning graph representations
with global structural information**
S. Cao et al.

Word2vec for node embedding cannot distinguish between $\circ \circ \circ$ and $\circ \circ \circ$ (the contexts are the same): replace the center word and the context with the first and last vertices of a length- k random walk; setting the gradient of the loss function to zero yields a matrix factorization problem,

$$W_i C_j = \log \frac{(A^k)_{ij}}{(A^k)_{\cdot j}} - \log \beta$$

(replace negative values with zero, to reduce noise); and concatenate all k -step representations, $k \in \llbracket 1, K \rrbracket$.

**New perspectives and methods
in link prediction**
R.N. Lichtenwalter et al. (2010)

Use *ensembles* of classifiers to reduce variance (C45, J48, naive Bayes, random forests). Try the following features:

- A variant of rooted PageRank: the probability that a random walk with no duplicates, starting at x , visits y in at most k steps;

- The maximum flow that can travel from x to y in at most 5 steps.

Computationally efficient link prediction in a variety of social networks

M. Fire et al. (2013)

Use a decision tree (J48) with node features (degree, Katz, number of paths of length 3, shared communities, etc.) to forecast missing links.

Link prediction with personalized social influence

Z. Huo et al. (2018)

Given a directed social network (e.g., Twitter), prune it to keep only significant connections, measured by the *mutual information* between the timing of user j 's tweets and the timing of user i 's retweets of user j 's tweets, and find node embeddings S and T (source and target) to model the link probability as

$$P[i \rightarrow j] = \frac{e^{S_i' T_j}}{\sum_k e^{S_i' T_k}}.$$

Link prediction in social networks: the state of the art

P. Wang et al. (2015)

Broken English

Connected patterns inspire link prediction in complex networks

M.Y. Zhou et al. (2017)

Do not use a raw node similarity score to predict missing links: learn a nonlinear transformation of the score. [Better: learn a nonlinear way of combining several of those scores.]

Link prediction via matrix factorization

A.K. Menon et al. (2011)

Model the link probabilities as

$$P[i \rightarrow j] = \sigma(u_i' \Lambda u_j + b_i + b_j + w' z_{ij} + x_i' V x_j)$$

where σ is the logistic function, $u_i' \Lambda u_j$ is a matrix factorization, x_i are the node features, w_{ij} the edge features and $x_i' V x_j$ a bilinear model (which works better than a linear one, $v' x_i + v' x_j$); add a regularizing penalty for U, V, Λ, W .

To overcome class imbalance, directly optimize the AUC by training on pairs of positive and negative samples – indeed, $\text{AUC} = P[f(i) > f(j) \mid y_i = 1, y_j = 0]$ (either arbitrary pairs, or pairs that share a node).

Link prediction based on graph neural networks

M. Zhang and Y. Chen (2018)

Graph neural networks can approximate node similarity measures and forecast missing edges.

Link prediction through deep learning

X.W. Wang et al. (2018)

Consider the adjacency matrix of a network as an image; reorder the nodes to make patterns more visible, using

- (Multiresolution) community detection;
- Consensus-based relabeling, which minimizes

$$\sum_i \frac{1}{k_i} \sum_j |r_{i,j+1} - r_{i,j} - 1|$$

where r_{ij} is the index of the j th neighbour of node i ;

- Or any matrix reordering algorithm;

apply a deep generative model (convolutional VAE or GAN – the latter works better) on perturbed copies of the adjacency matrix (obtained by removing 5% of the edges).

Link prediction: fair and effective evaluation

R. Lichtenwalter and N.V. Chawla

When the data is unbalanced, prefer the area under the precision-recall curve to that under the TPR-FPR one (ROC).

Evaluating link prediction methods

Y. Yang et al. (2013)

Idem

Link propagation: a fast semi-supervised learning algorithm for link prediction

H. Kashima et al.

Given a node similarity matrix W and a matrix indicating the presence (+1) or absence (−1) of edges F^* (unknown edges are left at 0, and the weights can be $|E|/|E^+|$ and $-|E|/|E^-|$ if the classes are unbalanced), **link propagation** completes the graph by minimizing

$$\sum_{ijk\ell} w_{ijk\ell} (f_{ij} - f_{k\ell})^2 + \lambda \sum_{(ij) \in E} (f_{ij} - f_{ij}^*)^2 + \mu \sum_{(ij) \notin E} f_{ij}^2$$

where the edge similarities are $W \otimes W$; this can be generalized to multiple edge types (we then have order-3 tensors), expressed with linear algebra operations, and optimized with gradient descent.

Link prediction in very large directed graphs: exploiting hierarchical properties in parallel

D. Garcia-Gasulla and U. Cortés

To infer missing links in directed graphs, generalize the neighbourhood-based similarity measures by separating ancestors and descendants, $N(x) = A(x) \sqcup D(x)$, e.g.,

$$\frac{\|A(x) \cap D(y)\|}{\|A(x)\|} + \frac{\|D(x) \cup D(y)\|}{\|D(x)\|}.$$

**Modeling relational data
with graph convolutional networks**
M. Schlichtkrull et al.

Graph convolutional networks (GCN) can be augmented with node and edge attributes or *types* (relational GCN).

**Graph convolutional neural networks
for web-scale recommender systems**
R. Ying et al.

To scale graph convolutional networks to large graphs, use samples of the neighbourhoods instead of the whole Laplacian.

Probabilistic relational models
L. Getoor et al.

A **probabilistic relational model** (PRM) is a *template* for Bayesian networks, linking node *types* rather than nodes (it can be seen as a Bayesian network on a graph with typed nodes, and parameters (or Bayesian priors on those parameters) shared among all nodes of the same type). Links can also be probabilistic and modeled from node attributes.

**Probabilistic entity-relationship models,
PRMs and plate models**
D. Heckman et al.

Probabilistic relational models (PRM) probabilistic entity-relation (ER) models (DAPER – contrary to PRMs, they distinguish between entities and relations) and plate models are equivalent.

Relational learning with Gaussian processes
W. Chu et al.

Model links as $P[i-j] = \sigma(f(x_i)'f(x_j)\varepsilon_{ij})$, where i, j are nodes, $f \sim \text{GP}$, $\varepsilon \in \{\pm 1\}$ and estimate $P[\varepsilon | \text{links}]$.

**Stochastic relational models
for discriminative link prediction**
K. Yu et al.

In a user-movie recommendation problem, the ratings are a function $\text{Users} \times \text{Movies} \rightarrow \mathbf{R}$ which can be modeled by a *tensor Gaussian process*, i.e., a Gaussian process whose kernel is a Kronecker product $\Sigma_{\text{Users}} \otimes \Sigma_{\text{Movies}}$.

**Dependency networks for inference,
collaborative filtering and data visualization**
D. Heckerman et al. (2000)

Dependency networks, in which the parents of a node are its Markov blanket,

$$\forall Y \notin \text{pa}(X) \quad X \perp\!\!\!\perp Y \mid \text{pa}(X),$$

are a potentially cyclic alternative to Bayesian networks, which do not lead to causal misinterpretations.

**Discriminative probabilistic models
for relational data**
B. Taskar et al.

Undirected graphical model template, i.e., graphical model with shared parameters.

**BPR: Bayesian personalized ranking
from implicit feedback**
S. Rendle et al. (2009)

The classification problem with three types of items, known positives, unknown positives and unknown negatives, is sometimes tackled by learning to separate known positives from the rest. Instead, one can look for a total order such that known positives dominate unknown items. This is similar to AUC optimization, but with the Heaviside objective

$$\sum_{\substack{i \text{ positive} \\ j \text{ unknown}}} \mathbf{1}_{x_i(\theta) > x_j(\theta)}$$

replaced with a differentiable (penalized) likelihood

$$\sum_{\substack{i \text{ positive} \\ j \text{ unknown}}} \log \sigma(x_i(\theta) - x_j(\theta)) + \lambda \|\theta\|^2,$$

corresponding to a logistic model

$$P[i > j] = \sigma(x_i(\theta) - x_j(\theta)).$$

For the item recommendation problem (one more index, for users, x_{ui}), the model can be a low-rank matrix factorization $X = WH'$ or an adaptive k -nearest neighbour model,

$$x_{ui} = \sum_{\substack{u \text{ likes } \ell \\ \ell \neq i}} c_{i\ell}, \quad C = HH'.$$

**Optimizing area under the ROC curve
using gradient descent**
A. Herschtal and B. Raskutti (2004)

Replace the Heaviside loss $\mathbf{1}_{x_i(\theta) > x_j(\theta)}$ with a differentiable approximation $\sigma(x_i(\theta) - x_j(\theta))$.

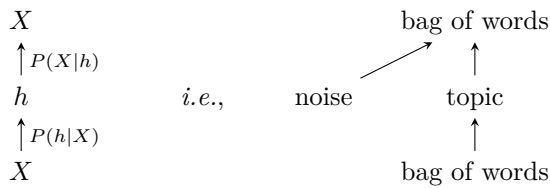
Naive Bayes for regression
E. Frank et al. (1999)

Naive Bayes does not work well for regression, unless there is really a lot of data.

$$\begin{aligned} P(Y|X_1, \dots, X_n) &\propto P(X_1, \dots, X_n, Y) \\ &= P(X_1|X_2 \dots)P(X_2|X_3 \dots) \dots P(Y) \\ &\approx P(X_1|Y) \dots P(X_n|Y)P(Y) \end{aligned}$$

**Neural variational inference
for text processing**
Y. Miao et al. (2016)

Variational autoencoders can model topics (neural variational document model).



**Discovering discrete latent topics
with neural variational inference**
Y. Miao et al. (2018)

Replace the Dirichlet prior for word distributions in LDA with neural-network-friendly distributions, e.g.,

- Gaussian softmax

$$x \sim N(0, I)$$

$$p = \text{softmax}(Ax)$$

- Gaussian stick-breaking ($\eta \sim \text{Beta}$ would give a Dirichlet distribution)

$$x \sim N(0, I)$$

$$\eta = \text{sigmoid}(Ax)$$

$$p_k = \eta_k \prod_{i=1}^{k-1} (1 - \eta_i)$$

$$p_n = \prod_{i=1}^n (1 - \eta_i)$$

- Recurrent stick-breaking

$$x \sim N(0, I)$$

$$\eta = \text{RNN}(x)$$

$$p = \text{StickBreaking}(\eta)$$

**TopicRNN: a recurrent neural network
with long-range semantic dependency**
A.B. Dieng et al.

Use a Gaussian distribution instead of a Dirichlet prior for the topic distribution of each document.

Instead of modeling each topic as a distribution on words, use a *language model* parametrized by the topic vector (RNN).

Model stopwords (problematic, in topic models, because they do not carry any information) using a mixture between the topic model and a distribution of stopwords (from a list).

**Deep learning for assessing banks' distress
from news and numerical financial data**
P. Cerchiello et al.

Feed the 600-dimensional doc2vec embedding and 12 financial ratios to a binary classifier (with one 50-neuron hidden layer).

**Empath:
understanding topic signals in large-scale text**
E. Fast et al.

MIT-licensed replacement for LIWC, with 200 categories selected from ConceptNet and the words obtained using cosine similarity with a skipgram embedding trained on amateur fiction (Wattpad – fiction is better for sentiment) and manually reviewed.

**Automatic keyword extraction
from individual documents**
S. Rose et al. (2010)

To extract multi-word keywords from an isolated document (not in a corpus), the RAKE algorithm proceeds as follows:

- Split the text on punctuation and stopwords to get keyword candidates (but allow candidates with one stopword if they are frequent enough);
- For each word, compute the frequency (number of occurrences), degree (number of words in the same candidates) and degree/frequency ratio;
- Define the keyword score by summing the word scores.

Also check TextRank.

To find stopwords from (manually generated) keywords, look for words at the boundary of keywords and rarely inside.

Interpretable machine learning
C. Molnar (2018)

Interpretability is needed:

- The problems we are solving are not completely specified – we do not really want to optimize a single metric but several, and the model output is not the final goal, just an intermediate step before some decision;
- We want to convince stakeholders that the model is safe, robust unbiased and fair.

But interpretability may make the system gamable.

Interpretability methods can be:

- Intrinsic (how the model actually works) or post hoc (as human explanations are);
- Local (only valid for observations similar to a given one) or global.

The output can be:

- Some feature summary (importance, contribution, visualization);
- Model internals (weights);
- Data points (prototypes, counterfactuals);
- An interpretable model.

Good explanations should

- Be contrastive, *i.e.*, explain why we got an output and not another;

- Be short (1 to 3 reasons, even if the model is more complex);
- Highlight abnormal or unusual inputs (outliers);
- Be consistent with prior beliefs (e.g., monotonicity);
- Be general (apply to a large number of instances).

1. Interpretable models include linear models (with a sparsity constraint), logistic regression, decision trees, decision rules, regression trees, naive Bayes, k -nearest neighbours.

2. Model-agnostic methods include:

- Individual conditional expectation (ICE) plots,

$$f(x, x_2, \dots, x_n) \sim x, \quad \text{for } (x_1, \dots, x_n) \in \text{Data};$$

- Partial dependency plots (PDP),

$$E[f(X_1, \dots, X_n) | X_1 = x] \sim x,$$

i.e., the average of the ICE curves; it can also be used for pairs of variables;

- Friedman's H statistic looks for interactions by comparing the partial dependency functions of X_1, X_2, X_{12}, X_{-1} ,

$$\begin{aligned} \text{PD}_I(x) &= E[\hat{f}(X) | X_I = x_I] \\ H^2 &= \frac{E[\text{PD}_{12}(X) - \text{PD}_1(X) - \text{PD}_2(X)]}{E[\text{PD}_{12}(X)^2]} \\ H^2 &= \frac{E[\hat{f}(X) - \text{PD}_1(X) - \text{PD}_{-1}(X)]}{E[\hat{f}(X)^2]} \end{aligned}$$

- Feature importance: permute the feature's values, and check the drop in performance;
- Global surrogate model;
- Local surrogate model (LIME), e.g., with a sparse linear model:
 - For text, randomly remove some of the words;
 - For images, segment them into superpixels;
- Shapley values

$$\begin{aligned} \phi_j &= \sum_{\substack{I \subseteq [1,p] \\ j \notin I}} \binom{p}{|I|} (v(I \cup \{j\}) - v(I)) \\ v(I) &= E[\hat{f}(X) | X_I = x_I] - E[\hat{f}(X)]. \end{aligned}$$

3. Example-based explanations include

- Counterfactual explanations: smallest changes to the features to change the outcome to a predefined value;
- Adversarial examples: small, insignificant perturbations of the input which nonetheless change the output;
- MMD critic: greedily select *prototypes* to reduce the maximum mean discrepancy between prototypes and data,

$$\text{MMD}(X, Y) = E[\kappa(X, X) - 2\kappa(X, Y) + \kappa(Y, Y)],$$

where κ is an RBF kernel, then greedily select *criticisms* where the prototypes and

the data distributions differ, *i.e.*, maximizing $|\text{witness}(x, \text{prototypes}, \text{data})|$

$$\text{witness}(x, X, Y) = E[\kappa(x, X) - \kappa(x, Y)],$$

with a diversity-inducing submodular penalty

$$\log \det(\kappa(x_i, x_j))_{i,j} \in \text{criticism};$$

- Influential instances (DFBETA, Cook's distance, influence function).

Peeking inside the black box: visualizing statistical learning with plots of individual conditional expectation
A. Goldstein et al. (2014)

The partial dependency plot integrates out X_2, \dots, X_n ,

$$E_{X_2, \dots, X_n} [f(x_1, X_2, \dots, X_n)]$$

while the ICE plots fix x_2, \dots, x_n (one curve for each observation)

$$f(x_1, \dots, x_n) \sim x_1.$$

Visualizing the effects of predictor variables in black box supervised learning models
D.W. Apley

The *partial dependency plot*

$$E_{X_2} [\hat{f}(X_1, X_2) | X_1 = x_1]$$

uses the marginal distribution of X_2 , and estimates the predictor \hat{f} very far away from the data, where it is unlikely to extrapolate well.

The *marginal plot*

$$E_{X_2 | X_1 = x_1} [\hat{f}(X_1, X_2) | X_1 = x_1]$$

uses the conditional distribution $X_2 | X_1 = x_1$ instead, but it suffers from the omitted variable bias: if X_2 plays a role, its impact will be included in the model intercept.

The *accumulated local effects* (ALE) plot

$$\int_{-\infty}^{x_1} E_{X_2 | X_1 = x_1} \left[\frac{\partial f}{\partial x_1}(X_1, X_2) \mid X_1 = x \right] dx$$

has the same interpretation without bias. For non-differentiable models (e.g., trees), use differences and sums instead of derivatives and integrals.

A unified approach to interpreting model predictions
S.M. Lundberg and S.I. Lee (2017)

Shapley values estimate all models on a subset of the variables $I \subset S$ and compares those with and without variable i .

***The separation plot: a new visual method
for evaluating the fit of binary models***
B. Greenhill et al. (2011)

To assess the quality of a logistic model, look at the ROC curve, the AUC, the Brier score

$$\frac{1}{n} \sum (\hat{p}_i - X_i)^2,$$

the pseudo- R^2 (likelihood ratio), the expected proportion of correct predictions

$$\text{EPCP} = \frac{1}{n} \left[\sum_{y_1=1} \hat{p}_i + \sum_{y_1=0} (1 - \hat{p}_i) \right],$$

the *separation plot* (1-dimensional tile plot, with one colour for each outcome, the cells ordered by the forecast, with the forecasted probability curve overlaid, and the expected number of events $\sum \hat{p}_i$ highlighted).

Deep learning: a critical appraisal
G. Marcus

Deep learning

- Needs infinite data;
- Cannot leverage background data, integrate prior knowledge, or learn abstractions from explicit verbal descriptions;
- Has limited transfer learning capabilities;
- Struggles with hierarchical structures;
- Is a black box;
- Cannot distinguish causation from correlation;
- Assumes the world is stationary;
- Has (frightening) non-human “failure modes”.

We need more unsupervised learning and hybrid symbolic-statistical systems.

***Understanding and simplifying
one-shot architecture search***
G. Bender et al. (2018)

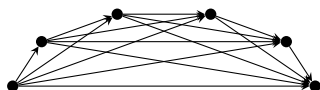
Learn a single large model, from which all the models to test can be obtained (an exponentially large number of them) by switching off some of the connections, à la dropout.

***Efficient neural architecture search
via parameter sharing***
H. Pham et al.

Search for an optimal subgraph in a large computational graph, seen as an ensemble of models sharing parameters.

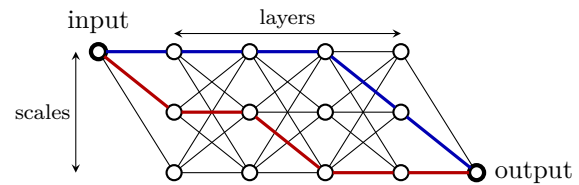
***You only search once:
single-shot neural architecture search
via direct sparse optimization***
X. Zhang et al. (2019)

Learn a complete DAG with a sparsity penalty.



Convolutional neural fabrics
S. Saxena and J. Verbeek

To choose the hyperparameters of a CNN, represent possible architectures as *paths* in a large network (dropout uses subgraphs instead of paths).



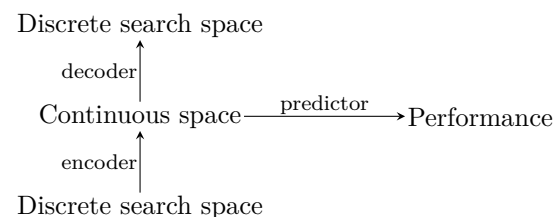
***FBNet:
hardware-aware efficient ConvNet design
via differentiable neural architecture search***
B. Wu et al.

Consider a “supernet” whose operators execute stochastically and look for the optimal architecture as a *distribution* (rather than a subnet).

***Mixed-precision quantization of convnets
via differentiable neural architecture search***
B. Wu et al.

Neural architecture optimization
R. Luo et al. (2018)

Use an autoencoder to map the discrete search space to a continuous one and predict the performance of an architecture from that continuous representation.



***Neural architecture search with
Bayesian optimization and optimal transport***
K. Kandasamy et al. (2018)

Gaussian-process-based Bayesian optimization can be used for network architecture search, with optimal transport to define a kernel to compare architectures (align the two networks and add penalties for mismatches) and an evolutionary algorithm to optimize the acquisition function.

Neural architecture search: a survey
T. Elsken et al. (2018)

**Neural architecture search
with reinforcement learning**
B. Zoph and Q.V. Le (2016)

Visualizing the loss landscape of neural nets
H. Li et al.

Plot the loss landscape of a neural net using one or two random directions in weight space, after *filter normalization* (normalize the weights in each filter (or each layer, for non-CNNs) to make them comparable – BatchNorm or ReLU units make the network scale-invariant).

- Small batch sizes lead to wider minima and better generalization;
- The loss landscape has a central convex region around the minimum, but is chaotic and non-convex beyond it – it is important to initialize the network in the convex region;
- Skip connections, small batches and wide networks (many filters) enlarge this region;
- The optimization path is very low-dimensional: use PCA rather than random dimensions to examine it.

Multi-dimensional graph Fourier transform
T. Kurokawa et al.

The **graph Fourier transform** (GFT) decomposes a signal $f : V \rightarrow \mathbf{R}$ on the eigenvectors of the Laplacian matrix $L = D - W$. It can be generalized to *cartesian product graphs* (e.g., time series for sensor networks, which have space and time dimensions), for which eigenvalues tend to be multiple: the eigenvalues (eigenvector) of the Kronecker sum $L_1 \oplus L_2$ are the sums (products) of those of L_1 and L_2 . It can be used, for instance, for signal filtering, to remove some frequencies in some directions:

$$\hat{f}_{\text{out}}(\lambda_1, \lambda_2) = h(\lambda_1, \lambda_2) \hat{f}_{\text{in}}(\lambda_1, \lambda_2).$$

**PassGAN: a deep learning approach
for password guessing**
B. Hitaj et al.

GAN to learn the distribution of real passwords from password leaks, to replace or complement rule-based password crackers (hashcat, johntheripper).

**WESPE: Weakly supervised photo enhancer
for digital cameras**
A. Ignatov et al.

GAN-based style transfer for photo enhancement, trained with highly liked Flickr images.

Multi-dimensional sparse super-resolution
C. Poon and G. Peyré (2017)

The **total variation norm** of a measure μ ,

$$\sup_{\substack{\eta \in \mathcal{C}(X) \\ \|\eta\|_{L^\infty} \leq 1}} \int_X \eta(x) d\mu(x)$$

generalizes the ℓ^1 and L^1 norms of vectors and functions. The corresponding penalty (BLASSO) can be used for *sparse superresolution*, i.e., to recover (position and amplitude of) spikes.

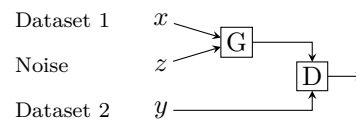
**StarGAN:
unified generative adversarial networks
for multi-domain image-to-image translation**
Y. Choi et al.

To translate images between domains (gender, hair colour, mood, age, etc.), use a single (conditional) generator $G : (\text{image}, \text{domain}) \mapsto \text{image}$, separate discriminators for the source and target domains, and a cycle loss to preserve image contents.

Datasets: CelebA (40 labels), RaFD (8 labels).

**DeblurGAN: blind motion deblurring using
conditional adversarial networks**
O. Kupyn et al.

To remove motion blur, use a Wasserstein **conditional GAN**



with *gradient penalty* and *perceptual loss* (L^2 norm of the difference of CNN features).

Improved training of Wasserstein GANs
I. Gulrajani et al.

Replace the weight clipping used to make the WGAN critic Lipschitz with a penalty on the norm of its gradient.

Non-local neural networks
X. Wang et al.

Self-attention, for images, is a generalization of non-local means,

$$y_i \propto \sum_j w(x_i, x_j) g(x_j)$$

with, e.g., $w(x_i, x_j) = \exp(Ax_i)'(Bx_j)$.

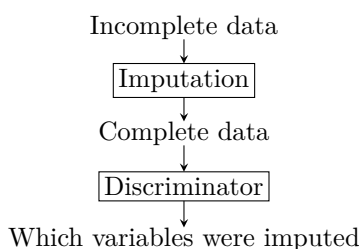
**Constructing unrestricted adversarial
examples with generative models**
Y. Song et al. (2018)

Train a GAN to model the class-conditioned distribution over inputs, then search its latent space (conditional on the desired class) for an adversarial example (of the desired class according to the GAN, but misclassified by the target classifier).

MolGAN: an implicit generative model for small molecular graphs
N. De Cao and T. Kipf

Use a GAN (WGAN with gradient penalty) to generate molecules (a vector for the atom types, a matrix for the link types, for a fixed number of atoms) and add a loss to force the generator to output molecules with desirable properties (e.g., easy to synthesize, using the RDKit cheminformatics software).

GAIN: missing data imputation using generative adversarial nets
J. Yoon et al. (2018)



Twin networks: matching the future for sequence generation
D. Serdyuk et al. (2018)

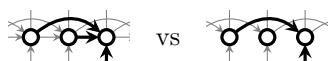
To encourage planning in a RNN, train (independently) a forward and a backward RNN to the same task, and add a penalty to match their hidden states.

Fraternal dropout
K. Źołna et al. (2018)

Train two identical copies of a RNN, with shared parameters, with different dropout masks, while minimizing the difference between their outputs.

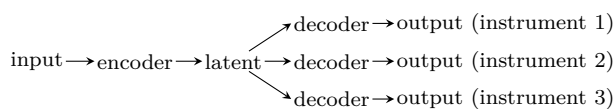
Dilated recurrent neural networks
S. Chang et al. (2017)

Slight variant on the skip connection.



A universal music translation network
N. Mor et al.

To change the instrument or style of a piece of music, use a WaveNet autoencoder, with one encoder and one decoder per style.



EraseReLU: A simple way to ease the training of deep convolutional networks
X. Dong et al.

There might be too many nonlinearities in some neural nets: removing the last ReLU of all basic modules may improve performance.

Shifting mean activation towards zero with bipolar activation functions
L.H. Eidner and A. Nøkland (2018)

Replace the nonlinearity $x \mapsto \sigma(x)$ with $x \mapsto (\sigma(x), -\sigma(-x))$ or with

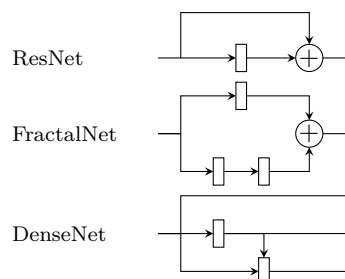
$$x_i \mapsto \begin{cases} \sigma(x_i) & \text{if } i \equiv 1 \pmod{2} \\ -\sigma(-x_i) & \text{if } i \equiv 0 \pmod{2} \end{cases}$$

The *orthogonal partition linear unit* (OPLU) partitions the coordinates into pairs and sorts them.

$$(x_i)_{1 \leq i \leq 2n} \mapsto \begin{pmatrix} \text{Max}(x_{2i-1}, x_{2i}) \\ \text{Min}(x_{2i-1}, x_{2i}) \end{pmatrix}_{1 \leq i \leq n}$$

SMASH: one-shot model architecture search through hypernetworks
A. Brock et al.

Encode neural network topologies as binary vectors, from a few building blocks (ResNet, FractalNet, etc.), and map them to weights.



DARTS: differentiable architecture search
H. Liu et al.

Search for computation cells with two inputs (previous two layers in a non-RNN, input and previous state in an RNN) by modeling them as DAGs on n nodes (n fixed), each connected to the previous ones, where the edges implement a linear combination of simple operations (zero, convolution, max pooling, etc.). Alternate between gradient steps in the architecture and weight spaces; for each edge, keep the most likely operation.

Differentiable learning-to-normalize via switchable normalization
P. Luo et al.

Successive normalization layers need not be identical: try linear combinations of normalizations (batch, layer, channel) with learned weights.

***Efficient neural architecture search
with neural morphism***
H. Jin et al.

Search for a good network architecture “morphing” it, by adding or removing layers or skip-connections, by changing the layer sizes, etc. (training is fast if we start from the previous, trained architecture).

For Bayesian optimization with Gaussian processes, use a kernel of the form $\kappa(a, b) = \exp -d(a, b)$, for some measure of distance d between the candidate architectures.

Regularization learning networks
I. Shavitt and E. Segal

Use a different regularization coefficient for each weight and replace the loss $\text{Loss}(w, \lambda) = \text{Loss}[w] + \exp(\lambda)' |w|$ with

$$\text{Loss}[w - \eta \nabla_w \text{Loss}(w, \lambda)].$$

***AutoAugment:
learning augmentation policies from data***
E.D. Cubuk et al.

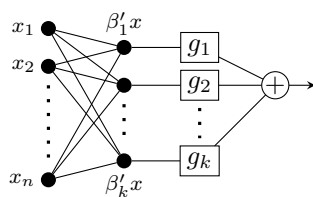
Parametrized data augmentation procedures can be learnt.

***Explainable neural networks
based on additive index models***
J. Vaughan et al. (2018)

Generalized additive models $f(x) = \sum f_i(x_i)$ and **additive index models**

$$f(x) = \sum g_i(\beta'_i x)$$

are interpretable and can be modeled by neural networks.



***Training deep autoencoders
for collaborative filtering***
O. Kuchaiev and B. Ginsburg

Use an autoencoder, with SELU units, tied weights (the decoder’s weights are the transpose of the encoder’s) and masked mean square error loss, for collaborative filtering. Since the input is sparse and the output $f(x)$ dense, re-feed the output to the autoencoder to have $f(f(x)) \approx f(x)$.

***A meta-learning perspective on cold-start
recommendations for items***
M. Vartak et al. (2017)

Meta-learning can replace matrix factorization for recommendation systems, when new items arrive continuously (tweet recommendation):

$$\text{user} \mapsto (\text{tweet} \mapsto \text{score}).$$

The meta-learning learns, for a user’s history (items liked or not), the weights of a neural network (either all the weights of a linear network, or the biases of a nonlinear one, the other weights being shared across users).

Tropical geometry of deep neural networks
L. Zhang et al.

Neural networks with ReLU activations are just tropical rational maps.

***Deep learning works in practice.
But does it work in theory?***
L.N. Hoang and R. Guerraoui (2018)

At equivalent Kolmogorov complexity, deeper neural networks compute functions with larger “non-parallelizable logical depth”.

***Generalizing Hamiltonian Monte Carlo
with neural networks***
D. Levy et al. (2018)

Neural networks can learn a reparametrization to speed up HMC, maximizing the expected squared jumped distance.

***Backpropagation through the void:
optimizing control variates
for black-box gradient optimization***
W. Grathwohl et al. (2018)

To minimize a function of the form

$$\text{Loss}(\theta) = \mathbb{E}_{b \sim p_\theta} [f(b)],$$

SGD needs gradients \hat{g} such that

$$E[\hat{g}] = \frac{\partial \text{Loss}}{\partial \theta}$$

and (ideally) $\text{Var} \hat{g}$ small. Common choices include

$$\begin{aligned} \hat{g}_{\text{reinforce}} &= f(b) \frac{\partial}{\partial \theta} \log p_\theta(b) \\ \hat{h}_{\text{reparam}} &= \frac{\partial f}{\partial b} \frac{\partial b}{\partial \theta}, \quad \text{where } b = b(\theta, \varepsilon), \varepsilon \sim p_\varepsilon. \end{aligned}$$

Control variates can reduce variance:

$$\hat{g}_{\text{new}}(b) = \hat{g}(b) - c(b) + \mathbb{E}_{b \sim p_\theta} [c(b)].$$

They can also deal with non-differentiable f and discrete b ,

$$\begin{aligned}\hat{g}_{\text{lax}} &= \hat{g}_{\text{reinforce}}[f] - \hat{g}_{\text{reinforce}}[c_\phi] + \hat{g}_{\text{reparam}}[c_\phi] \\ &= [f(b) - c_\phi(b)] \frac{\partial \log p_\theta(bv)}{\partial \theta} + \frac{\partial c_\phi(b)}{\partial \theta}\end{aligned}$$

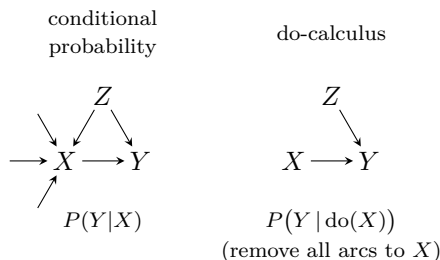
where c_ϕ is a neural network approximating f .

$$\begin{aligned}\hat{g}_{\text{DLAX}} &= f(b) \frac{\partial}{\partial \theta} \log p_\theta(b) - c_\phi(z) \frac{\partial}{\partial \theta} \log p_\theta(z) + \frac{\partial}{\partial \theta} c_\phi(z) \\ \hat{g}_{\text{RELAX}} &= [f(b) - c_\phi(\tilde{z})] \frac{\partial}{\partial \theta} \log p_\theta(b) + \frac{\partial}{\partial \theta} c_\phi(z) - \frac{\partial}{\partial \theta} c_\phi(\tilde{z}) \\ b &= H(z), \quad z \sim p_\theta(z), \quad \tilde{z} \sim p_\theta(z|b)\end{aligned}$$

Beyond power calculations: assessing type S (sign) and type M (magnitude) errors **A. Gelman and J. Carlin (2014)**

Besides the p -value and the *power*, also consider the *type S error rate*, *i.e.*, the probability that the sign is incorrect given that the parameter is significantly different from zero, and the *exaggeration ratio*, *i.e.*, the expectation of the absolute value of the estimate over the effect size, given it is significantly different from zero. (R code provided)

Implicit causal models for genome-wide association studies **D. Tran and D.M. Blei**



How many random seeds? Statistical power analysis in deep reinforcement learning experiments **C. Colas et al.**

The significance (often, $\alpha = 0.05$) of a statistical test only controls the type I errors. Compute the sample size needed to achieve the desired power (e.g., type II error $\beta = 0.20$), for the desired effect size ε , and the estimated variances.

In practice, to compare deep learning or reinforcement learning algorithms, run them at least 20 times with different random seeds.

Measuring scientific broadness **T. Price and S. Hossenfelder**

To measure the “broadness” of a researcher:

- Retrieve Arxiv abstracts using the API;

- Extract “keywords”: sequences of at most 10 words appearing in at least 20 papers; clean them (remove s’s, spaces; stem);
- Compute the “rank” of the keywords:

$$\text{KL}(P[\text{category}|\text{keyword} = k] \parallel P[\text{category}]) \times \left(1 - \exp - \frac{P[\text{keyword}]}{r}\right)$$

- Estimate an LDA model on the top 40,000 keywords (50 topics, 5 passes);
- Define the “broadness” of an author as the entropy of the topic distribution of her papers.

A network approach to topic models **A. Gerlach et al.**

Topic modeling is similar to community detection on the bipartite graph of documents and words. The *stochastic block model* (DBM) is equivalent to pLSI (frequentist LDA, *i.e.*, LDA without a prior), but has too many parameters to be used without a prior. The *hierarchical SBM* provides such a prior and removes the unimodality assumption inherent in LDA.

Implementation in `graph-tool` and `TopSBM`.

A high-reproducibility and high-accuracy method for automated topic classification **A. Lancichinetti et al.**

TopicMapping is an initialization method for LDA:

- Eliminate non-discriminative words by looking at all the pairwise cosine similarities;
- Run a clustering (community detection) algorithm, e.g., InfoMap, on the bipartite graph of words and documents (if the vocabularies used by the topics were disjoint, the topics would be the connected components);
- Use the clusters to estimate a PLSA model;
- Use this model as a starting point to estimate an *asymmetric LDA* model (LDA with a penalty for topic distribution entropy).

Robust PLSA performs better than LDA **A. Potapenko and K. Vorontsov (2013)**

Variants of LDA can be estimated in the same way:

- LDA uses a Dirichlet prior and defines topics as distributions on words and documents as mixtures of topics;
- PLSA replaces the Bayesian priors with maximum likelihood estimation (the priors can be seen as a penalty);
- SWB (specific words and background) is similar, but models documents as a mixture of topics, a corpus-wide background word distribution, and a document-specific distribution.

The map equation
R. Rosvall et al. (2009)

Infomap looks for a community decomposition of a graph minimizing the Huffman encoding of a random walk using community×node pairs. The code length is easy to compute.

$$\begin{aligned}\alpha &: \text{node} \\ p_\alpha &: \text{node visit probability} \\ i &: \text{community} \\ q_{i\curvearrowright} &: \text{exit probability} \\ q_{\curvearrowright} &= \sum_i q_{i\curvearrowright} \\ p_{i\cup} &= q_{i\curvearrowright} + \sum_{\alpha \in i} p_\alpha \\ L(M) &= q_{\curvearrowright} H(Q) + \sum_i p_{i\cup} H(P_i)\end{aligned}$$

To minimize it, start with each node in its own community; at each iteration, move a random node to the community decreasing the description length the most; when nothing can be improved, collapse each community into a node, and start again.

**Community detection and visualization
of networks with the map equation framework**
L. Bohlin et al.

Incorporating lexical priors into topic models
J. Jagarlamudi et al.

To force some (rarer) topics to appear, model each topic as a mixture of a regular topic (a distribution on words) and a seed topic, *i.e.*, a distribution on a user-specified set of words (SeededLDA).

**Paper abstract writing
through editing mechanism**
Q. Wang et al.

Use seq2seq with attention to generate an abstract draft from a title, and then a final abstract from the draft.

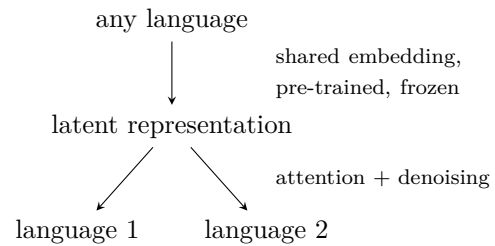
A simple method for commonsense reasoning
T.H. Trinh and Q.V. Le

To disambiguate pronouns, substitute them with all possible candidates and use a language model (trained on a massive amount of unlabeled data) to choose the most probable sentence.

Unsupervised neural machine translation
M. Artetxe et al. (2018)

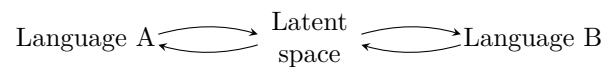
Use a pre-trained, frozen, shared embedding and attention-based denoising decoders, trained with back-

translation.



**Unsupervised machine translation
using monolingual corpora only**
G. Lample et al. (2018)

Use denoising autoencoders.



Word translation without parallel data
A. Conneau et al. (2018)

To align word embeddings for two different languages, given a dictionary of 5000 pairs of words, find an orthogonal transformation W that maps one to the other (Procrustes problem),

$$W^* = \underset{W \in O}{\operatorname{Argmin}} \|WX - Y\|_F = UV',$$

where $YX' = U\Sigma V'$ is the SVD.

With no bilingual dictionary, learn an orthogonal transformation W and a discriminator to distinguish a random sample from $\{Wx_1, \dots, Wx_n\}$ from one from $\{y_1, \dots, y_m\}$. The orthogonality constraint can be enforced by alternating the gradient updates with

$$W \leftarrow (1 + \beta)W - \beta WW'W, \quad \beta = 10^{-2}.$$

The alignment can be improved by selecting the most frequent words, assuming their nearest neighbour is the correct translation and using the Procrustes method.

The notion of nearest neighbour is not symmetric: some points are the nearest neighbour of many points (hubs), others of no points (anti-hubs).

The similarity between source and target words can be measured as

$$\text{CSLS}(Wx, y) = 2 \cos(Wx, y) - \langle \cos(Wx, \cdot) \rangle - \langle \cos(\cdot, y) \rangle,$$

where the average cosine similarities are computed over neighbourhoods of Wx and y in the bipartite graph of k -nearest neighbours.

On the tree-likeness of hyperbolic spaces

M. Hamann

An \mathbf{R} -tree is a topological space X such that there exists a unique arc between any two points.

Two geodesic rays (unbounded geodesics, isometric images of $(0, \infty)$), π_1 , π_2 are equivalent if, for all sequences $(x_n)_n$ in π_1 , there exists M such that $\lim d(x_n, \pi_2) \leq M$. The *hyperbolic boundary* ∂X of X is the set of equivalence classes of geodesic rays. There are geodesic (double) rays between boundary points and between boundary points and interior points.

Given a proper hyperbolic space X (the closed balls are compact and there is a geodesic between any two points), there exists a topological \mathbf{R} -tree T whose rays are quasi-geodesics and such that every geodesic ray in X lies eventually close to a ray of T ; the embedding $T \hookrightarrow X$ extends to $\partial T \hookrightarrow \partial X$ (no longer an embedding).

The *Assouad dimension* is defined by

$$S(\alpha, \beta) = \{\#V : \forall x \neq y \in V \quad \alpha \leq d(x, y) \leq \beta\}$$

$$\dim_A(X) = \inf\{s \geq 0 : \forall 0 < \alpha \leq \beta \quad \exists c \geq 0$$

$$S(\alpha, \beta) \leq c(\beta/\alpha)^s\}.$$

Hyperbolic deep learning for Chinese natural language understanding

M.V. Micic and H. Chu

The *hyperbolic skipgram* character embedding replaces the Euclidean product with the Lorentzian product in the hyperboloid model $\mathbf{H}^n \subset \mathbf{R}^{(n,1)}$

$$\langle x, y \rangle_{\mathcal{L}} = \sum_{i=1}^n x_i y_i - x_{n+1} y_{n+1}$$

$$\mathbf{H}^n = \{x \in \mathbf{R}^{(n,1)} : \langle x, x \rangle_{\mathcal{L}} = -1 \text{ and } x_{n+1} > 0\}$$

(with an additive shift $\langle x, y \rangle \rightsquigarrow \langle x, y \rangle_{\mathcal{L}} + \theta$).

Transform the character embedding from the hyperboloid model to the Poincaré model and apply a *hyperbolic transformer* to an *intent classification* dataset (text-based commands to an Alexa-like machine).

Train the character embedding on the linguistic data consortium corpus and the word embedding (form comparison) on the People's daily corpus (both commercial).

Analytic hyperbolic geometry defines a scalar product and a (non-commutative, non-associative) sum between points in a hyperbolic space (*gyrovectors*):

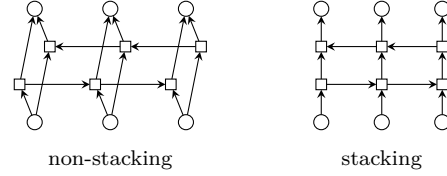
$$\lambda \otimes x = \exp_0(\lambda \log_0 x)$$

$$x \oplus y = \exp_x(P_{0 \rightarrow x} \log_0 y)$$

State-of-the-art Chinese word segmentation with bi-LSTMs

J. Ma et al.

Stacked biLSTM, with characters and bigrams as inputs, pre-trained embeddings, dropout (for the LSTM parameters) and hyperparameter tuning; data from the Chinese Penn treebank, Chinese universal treebank and SIGHAN2005.



Long short-term memory neural networks for Chinese word segmentation

X. Chen et al.

1- or 2-layer LSTM, with character of n -grams as inputs, and character embeddings.

Hyperbolic neural networks

O.E. Ganea et al.

Logistic regression

$$P[y = 1 | x] \propto \exp(\langle a, x \rangle - b)$$

can be generalized to hyperbolic space by noticing

$$\langle a, x \rangle - b = \text{sign}(\langle a, x \rangle - b) \|a\| d(x, H_{a,b})$$

$$= \text{sign}(\langle -p + x, a \rangle) \|a\| d(x, H_{a,b})$$

where $b = \langle a, p \rangle$ and

$$H_{a,b} = \{x : \langle a, x \rangle = b\}$$

$$H_{a,p} = \{x : \langle -p + x, a \rangle = 0\} = p + \{a\}^\perp$$

and replacing $+$ with \oplus .

The linear (and affine, non-linear) transformations $x \mapsto Ax$ used in neural networks can be generalized to hyperbolic space as $x \mapsto \exp_0(A \log_0 x)$.

There is no Riemannian Adam: use Riemannian SGD (projected SGD is less stable).

Hyperbolic attention networks

C. Gulcehre et al.

Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm

B. Felbo et al.

DeepMoji is a 2-layer LSTM network with attention trained to forecast the presence of 64 common emojis from the text of tweets; use it for transfer learning or provide sentiment-aware sentence embeddings.

Pretrained model available.

**PlusEmo2Vec at SemEval-2018 task 1:
exploiting emotion knowledge
from emoji and hashtags**
J.H. Park et al.

Use a biLSTM to forecast (emotion-related) emoji clusters. Use the resulting sentence representation to forecast emotions, using a training set built from hashtags (from a sentiment lexicon).

This gives an *emotional word embedding*, very different from the traditional ones (word2vec, glove), which struggle to distinguish sentiment – indeed, positive and negative words are used in the same context (e.g., “a good/bad movie”).

日本語単語の難易度推定の試み
Y. Mizutani et al.

Measure the difficulty of a word by that of its context, initially estimated from vocabulary lists for middle, junior high and senior high school.

What is... a spectrahedron?
C. Vinzant

A *spectrahedron* is a set of the form

$$\{x \in \mathbf{R}^n : A_0 + x_1 A_1 + \cdots + x_n A_n \succcurlyeq 0\}$$

where the A_i ’s are symmetric matrices; they generalize polytopes.

Semidefinite programming maximizes a linear function over a spectrahedron; it generalizes linear programming.

The *elliptope* is the spectrahedron of correlation matrices of size n .

Spectrahedra are not closed under projections.

**Fréchet distance based approach
for searching online handwritten documents**
R. Sriraghavendra et al.

The *Fréchet distance* between two curves P and Q is

$$\min_{\alpha, \beta} \max_{t \in [0, 1]} \|P \circ \alpha(t) - Q \circ \beta(t)\|_2$$

where the reparametrizations $\alpha, \beta : [0, 1] \rightarrow [0, 1]$ are nondecreasing and surjective. (Think of a man and his dog, walking along P and Q , never going backwards – the distance is the minimum leash length.)

The discrete Fréchet distance can be computed with dynamic programming.

The *prefix Fréchet distance* can be used to retrieve handwriting, *i.e.*, time series (t, x, y) .

R Journal (2018-2)

The generalized autoregressive score (**GAS**) model models time-varying distributions $p(\cdot, \theta_t)$ whose parameters θ_t follow an AR(1) process driven by the

score function

$$\nabla_t = \frac{\partial \log p(y, \theta)}{\partial \theta} \Big|_{y_t, \theta_t}$$

$$\theta_{t+1} = \kappa + A(\text{Var}_{t-1} \nabla_t)^{-\gamma} \nabla_t + B\theta_t, \quad \gamma \in \{0, \frac{1}{2}, 1\}.$$

The maximum likelihood estimation is straightforward.

The **dynamac** package estimates and tests **autoregressive distributed lag models** (ARDL)

$$y \text{ or } \Delta y \sim f(\text{time}) + x + \Delta x + \log(y) + \log(x) + \log(\Delta y) + \log(\Delta x),$$

where the series can be stationary or (co)integrated.

In a semi-Markov model, sojourn times are arbitrarily distributed (in a Markov model, they are exponential or geometric). Check **SMM** (discrete-time SMMs), **semiMarkov**, **hsmm**, **mhsmm**.

The **lmridge** package provides ridge regression diagnostics.

The **BNSP** and **bamlss** packages provide Bayesian GAMLSS with spike-and-slab priors for variable selection.

Differential item functioning looks for unfairness in item response theory (IRT) models: even if they have the same ability, different groups may respond differently to the same distractor.

The **nsROC** package provides confidence bands for the ROC curve (**ROCbands**) and compares ROC curves (**compareROCdep**). Also check **pROC**, **ROCR** (estimators, with smoothing, and confidence intervals for the AUC); **plotROC**; **fbroc** (bootstrap); etc.

The k -means algorithm, can be generalized to mixed data types (k -prototypes):

$$d(x, y) = \sum_{i \text{ continuous}} (x_i - y_i)^2 + \lambda \sum_{i \text{ discrete}} \mathbf{1}_{x_i \neq y_i}.$$

Check **clustMixType**, **gower**, **cluster** (**daisy**, **hclust**, **agnes**), **CluMix**, **flexclust** **fpc**, **clustMD**, **kamila**, **klar**.

Instead of forward or backward variable selection, the *feasible solutions algorithm* (FSA) starts with a random set of variables and replaces them, one at a time, to increase the criterion (AIC, etc.), until this is no longer possible (repeat several times with different starting points).

There are several implementations of FastICA in **fICA**, **fastICA**, **ica**.

The **testforDEP** package provides more independence tests, beyond correlation, rank correlation, Kendall’s τ and Hoeffding’s test (comparing $F_{X,Y}$ and $F_X F_Y$), not limited to linear or monotonic dependence.

The **clikcorr** estimates bivariate correlation with censored or missing data. The **revenge** package estimates the parameters of a Poisson or binomial distribution from censored data.

Two-dimensional density estimation is available in `kde2d`, `bkde2D` (binned – significantly faster), `pointdensityP`.

Uncertainty can be propagated using Monte Carlo simulations or Taylor expansions: `errors`, `propagate`, `metRology`.

The `wrapr` package generalizes `matrittr`'s pipe.

The `qqplotr` package provides confidence intervals for quantile plots (computed from normal approximation, a KS test, bootstrap, etc.).

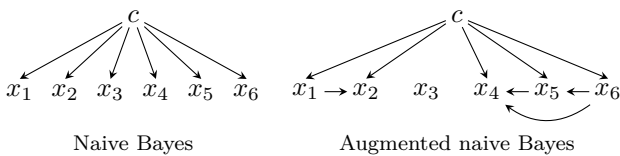
The `utilml` and `mldr` packages deal with *multilabel classification* (each observation can have several labels); the algorithms are built using decision trees, k -nearest neighbours, naive Bayes classifiers, support vector machines (SVM) or `xgboost`.

Many packages provide interpretations of complex, black-box models: `lime` (simple, interpretable model estimate on similar observations), `iml` (Shapley values, *i.e.*, contributions of a (local) linear model on binary variables), `DALEX`, `live` (locally interpretable visual explanations, *i.e.*, LIME after perturbing the instance one coordinate at a time), `breakDown` (additive contribution of each variable,

$$f(x, y) = \langle f(\cdot, \cdot) \rangle + \langle f(x, \cdot) \rangle - \langle f(\cdot, \cdot) \rangle + \langle f(x, y) \rangle - \langle f(x, \cdot) \rangle,$$

ordering the variables from the least to the most important, as measured by $\langle f(x, \cdot) \rangle - \langle f(\cdot, \cdot) \rangle$), `pdp` (partial dependency plots, ICE), `ALEPlot`, `FactorMerger`.

The `bnclassify` package learns (structure and parameters of) **Bayesian network classifiers (augmented naive Bayes models)**; it is similar to `bnlearn`, but focuses on $p(c|x)$ rather than $p(c, x)$, and drops variables irrelevant for the classification task.



Vertex bootstrap resamples the vertices of a graph, with replacement, and returns a new adjacency matrix; for duplicated vertices, since there are no self-edges, take a random element of the initial adjacency matrix. **Patchwork bootstrap** uses the subgraph of nodes at most n steps away from a random set of seed vertices (egonets); for bootstrap estimates of the degree distribution, adjustments are needed for the degrees of the vertices and the time(s) at which they enter the patch (use each directed edge exactly once). Check the `snowboot` package.

NetworkToolbox:
methods and measures for brain, cognitive

and psychometric network analysis in R
A.P. Christensen (R Journal 2018)

A correlation (or partial correlation) matrix can be turned into a graph using the *graphical lasso* (`bootnet`, `glasso`, `IsingFit`) or *information filtering* (maximum spanning tree, planar/triangulated maximally filtered graph (PMFG/TMFG)), *e.g.*,

- Start with the tetrahedron with the largest sum of weights;
- Add the node and 3 edges maximizing the sum of weights;
- Iterate until all the nodes have been added.

The (asymmetric) *dependency matrix* is

$$D(i, j) = \text{Mean}_{k \neq j} \text{Cor}(i, k) - \text{pCor}(i, k|j).$$

The *randomized shortest path betweenness centrality* has fewer outliers than the betweenness centrality. The *node impact* is the difference between the average shortest path length without and with it – if positive, its presence makes communities closer together. Community detection can be used as a replacement for PCA. Measures of centrality can also be computed within each community. Also check EGA.

Two betweenness centrality measures based on randomized shortest paths
I. Kivimäki et al. (2016)

The randomized shortest path betweenness is the expected number of visits of a node, for the probability distribution on paths between two given nodes

$$P[\text{path}] \propto \exp - \frac{\text{path length}}{\text{temperature}}.$$

UMAP: uniform manifold approximation and projection for dimension reduction
L. McInnes et al. (2018)

A *set* is a functor $* \rightarrow \mathbf{Set}$. A *fuzzy set* is a functor $([0, 1], \leq) \rightarrow \mathbf{Set}$ (the classical definition of fuzzy sets, as maps $X \rightarrow [0, 1]$ measuring “membership strength”, only provides subsets).

A *simplicial set* is a functor $\Delta^{\text{op}} \rightarrow \mathbf{Set}$. A *fuzzy simplicial set* is a functor $\Delta^{\text{op}} \rightarrow \mathbf{Fuzz}$.

Extended pseudometric spaces (they allow $d(x, y) = \infty$ and do not require $d(x, y) = 0 \implies x = y$) and non-expansive maps form a category \mathbf{EPMet} .

There is an adjunction between (finite) simplicial fuzzy sets and (finite) EPM spaces

$$\begin{array}{ccc} \mathbf{sFuzz} & & \\ \downarrow \text{metric realization} & \dashv & \uparrow \text{singular complex} \\ \mathbf{EPMet} & & \end{array}$$

which allows us to glue, as fuzzy simplicial sets, metric spaces that cannot be glued together in the category of metric spaces.

The UMAP algorithm computes the weighted k -nearest neighbour (the 1-skeleton of the fuzzy simplicial set obtained by glueing the local neighbourhoods of all points), with exponentially decaying weights (the weights are given by the adjunction), and applies a force-directed graph layout algorithm (the repulsive forces are only applied to a sampling of the vertices).

Efficient k -nearest neighbour graph construction for generic similarity measures
W. Dong et al. (2011)

The *nearest neighbour descent* algorithm builds an approximate k -nearest neighbour graph by starting with a random neighbourhood for each point and improving the estimate by looking at neighbours of neighbours.

Visualizing large-scale and high-dimensional data
J. Tang et al. (2016)

LargeViz computes an approximate k -nearest neighbour graph, using a few random projection trees (built from hyperplanes equidistant from two random points in each non-leaf node), refines it by looking at neighbours of neighbours, assigns t -SNE weights to the edges, and lays out the graph by maximizing

$$\prod_{(i,j) \in E} f(\|y_i - y_j\|)^{w_{ij}} \prod_{(i,j) \notin E} [1 - f(\|y_i - y_j\|)]^\gamma$$

using gradient descent and edge sampling (proportionally to the weights), with $f(x) = (1 + e^{x^2})^{-1}$ or $f(x) = (1 + ax^2)^{-1}$.

Homology-preserving dimensionality reduction via manifold landmarking and tearing
L. Yan et al. (2018)

Isomap reduces the dimension of a point cloud by computing its k -nearest neighbour (k -NN) graph and applying multidimensional scaling (MDS) to the geodesic distance.

L-Isomap (landmark isomap) only applies MDS to a set of (often randomly selected) “landmarks”; the other points are added using the geodesic distance to the landmarks.

The **Reeb graph** of a function $f : X \rightarrow \mathbf{R}$ from a manifold X is obtained by contracting each connected component of the level sets $f^{-1}(\{a\})$ to a point. From a cloud of points, use the inverse image of intervals $f^{-1}((a, b))$ instead (Mapper algorithm) and some clustering algorithm (e.g., DBSCAN) to approximate the connected components. The centroids of the components can be used as landmarks. The function f is often the distance to some fixed point.

Try to cut the K -NN graph with a plane orthogonal to one of the edges of the Reeb graph, and check if this preserves more of the topology by looking at the Wasserstein distance, or the bottleneck distance, between the persistence diagrams of the original point cloud and the dimension-reduced one.

A physical model for efficient ranking in networks
C. de Bacco et al.

To estimate a hierarchy (to assign a real-valued rank to each node) from a directed graph (in which most edges are top-down or between nodes of similar rank),

$$\begin{aligned} \text{Find} \quad & (s_i)_{i \in V(G)} \in \mathbf{R}^{|V(G)|} \\ \text{To minimize} \quad & \sum_{(i \rightarrow j) \in E(G)} (s_i - s_j - 1)^2 \end{aligned}$$

The social bow tie
H. Mattie et al.

Use the Jaccard similarity of the neighbourhoods of the ends of an edge as a measure of its “strength”.

Ctrl+Z: Recovering anonymized social graphs
Y. Zhang et al.

To anonymize graphs, one can add edges to ensure that nodes cannot be identified by their degree (there are at least k other nodes with the same degree), or add edges to add noise to the dk2 series (number of edges between nodes of degree i and j).

Word2vec on random walks on the graph provide a node embedding. The cosine similarity between the embeddings of the ends of an edge measures its plausibility, and can be used to deanonymize the graph.

Few-shot learning with graph neural networks
V. Garcia and J. Bruna (2018)

A **graph neural network** (GNN) takes an input signal on its vertices, learns a linear combination of local linear operators (e.g., the adjacency matrix and its first powers), and applies a non-linearity – the output is another weighted graph.

Message-passing algorithms suggest to also learn edge features, e.g., $\phi(x_i, x_j) = \text{MLP}(|x_i - x_j|)$ for $i \sim j$.

Towards gene expression convolutions using gene interaction graphs
F. Dutil et al.

To take gene interactions into account, use a *network-regularized sparse regression*, with penalty $\lambda |w|^L |w|$, where L is the graph Laplacian, or a graph convolution network.

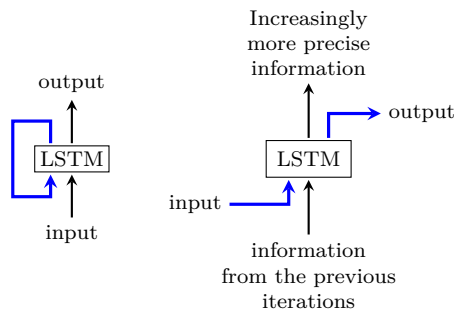
Graph convolutional neural networks for web-scale recommender systems
R. Ying et al. (2018)

To train graph convolutional networks (GCN) on large graphs, do not use the whole neighbourhood bonds, but use random walks to sample them (rather than random neighbours)

Feedback networks

A.R. Zamir et al.

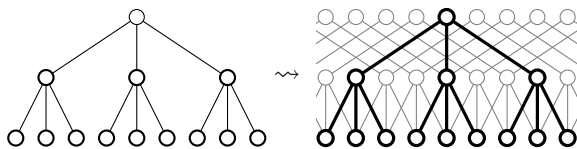
Add feedback in (otherwise non-recurrent) networks.



Dilated residual networks

F. Yu et al.

Remove striding and (to keep large receptive fields) use *dilated* convolutions. It is possible to remove the gridding artefacts by removing the remaining pooling layers and adding more convolutional layers, without residual convolutions, at the end.



Learning scalable deep kernels with recurrent structure

M. Al-Shedivat et al. (2017)

To get a deep Gaussian process (GP) kernel with a recurrent structure, transform the input with an LSTM and build a kernel in the transformed space. The loss function no longer factorizes over the data: either pre-train the network separately and fine-tune with full batches, or use semi-stochastic alternating gradient descent.

Unsupervised neural machine translation

M. Artetxe et al.

Denoising autoencoders can provide decent translations even in the absence of any crosslingual data.

language 1 \rightarrow language 2 \rightarrow language 1
language 2 \rightarrow language 1 \rightarrow language 2

Differentiable plasticity: training plastic neural networks with backpropagation

T. Miconi et al. (2018)

In a **Hebbian network**, connections between neurons that tend to fire together are reinforced. Allow for both baseline and *plastic* (Hebbian) behaviour,

$$\text{weight}_{ij} = w_{ij} + \alpha_{ij} \text{Hebb}_{ij}$$

where the *Hebbian trace*, reset at the beginning of each episode, (when switching to a new problem) is one of

$$\text{Hebb}_{ij}(t+1) = \text{EWMA}_{\eta}[x_i(t)x_j(t)]$$

$$\text{Hebb}_{ij}(t+1) = \text{EWMA}_{\eta}[x_i(t-1)x_j(t)]$$

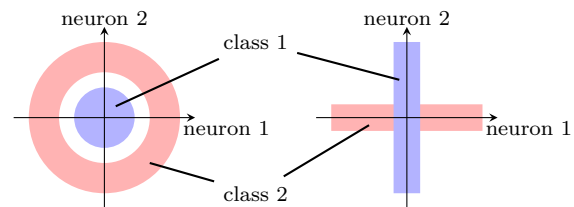
$$\text{Hebb}_{ij}(t+1) = \text{Hebb}_{ij}(t) + \eta x_j(t)[x_i(t-1) - x_j(t)\text{Hebb}_{ij}(t)]$$

and w_{ij} , α_{ij} and η are learnt by gradient descent.

Variance networks: when expectation does not meet your expectations

K. Neklyudov et al.

Stochastic neural nets (e.g., Bayesian neural nets) replace deterministic weights with probability distributions; often, at test time, only the means are used. Try a **variance layer**, *i.e.*, stochastic weights with zero mean and learned variances.



Tensor regression networks with various low-rank tensor approximations

X. Cao et al.

Constrain the weights of a layer to have low rank or low tensor rank by parametrizing them as a product of rectangular matrices, or as a tensor decomposition (CP, Tucker, TensorTrain).

Independently recurrent neural network (IndRNN): building a longer and deeper RNN

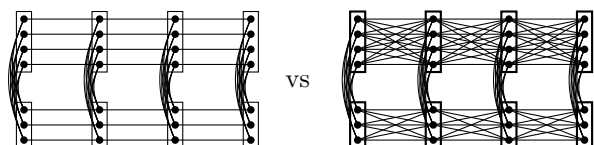
S. Li et al.

Replace the RNN updates

$$h_t = \sigma(Wx_t + Uh_{t-1} + b)$$

with

$$h_t = \sigma(Wx_t + u \odot h_{t-1} + b).$$



Max-Mahalanobis linear discriminant analysis networks

T. Pang et al. (2018)

The last layer of a neural net for classification is usually a (multi)logistic regression: try linear discriminant analysis (LDA) instead.

A flexible approach to automated RNN architecture generation
M. Schrimpf et al.

A DSL for automated RNN architecture search with reinforcement learning suggests an alternative (BC3) to LSTM and GRU cells.

Progressive neural architecture search
C. Liu et al.

Bayesian optimization (SMBO), on a hierarchical cell-based search space, to choose the structure of a CNN.

Learning to learn by gradient descent by gradient descent
A. Andrychowicz et al.

Momentum-like learning rules of the form

$$\theta_{t+1} \leftarrow \theta_t + \text{RNN}(\nabla f(\theta_t)),$$

where f is a random function, can be learnt by stochastic gradient descent. Use coordinate-wise LSTM units, with a few “global averaging cells”, whose output is averaged across all dimensions.

Neural optimizer search with reinforcement learning
I. Bello et al. (2017)

Idem.

Learning to optimize
K. Li and J. Malik

Gradient-descent-like optimization algorithms can be learnt by reinforcement learning (guided policy search): the state space contains the previous 25 gradients and improvements in the objective; the action is the step size.

Guided policy search
S. Levine and V. Koltun (2013)

The gradient of the policy can be estimated from sample trajectories (“likelihood ratio methods”). Importance sampling (with some regularization) allows off-policy learning. A guiding distribution can be obtained as the Gaussian i -projection of the Boltzmann distribution ρ on trajectories ζ with the negative reward as energy,

$$\underset{q \text{ Gaussian}}{\text{Argmin}} \text{KL}(q \parallel \rho) = \underset{q \text{ Gaussian}}{\text{Argmin}} \mathbb{E}_{\zeta \sim q} [-r(\zeta)] - H(q)$$

twisted with the current policy $\bar{r}(x, a) = r(x, a) + \log \pi_\theta(a|x)$.

Learning neural network policies with guided policy search under unknown dynamics
S. Levine and P. Abbeel

Sample trajectories from the current policy; use them to compute a linear Gaussian approximation of the model; compute the corresponding optimal policy (“iterative linear Gaussian regulator”, iLGR), but constrain them to be close (for the KL divergence) to the previous ones.

An intriguing failing of convolutional neural networks and the CoordConv solution
R. Liu et al.

Add the coordinates to the inputs of a convolutional layer.

IGLOO: slicing the feature space to represent long sequences
V. Sourkov

To achieve long-term memory with a CNN, compress the output of a convolutional layer with *sparse reservoir computing* (L times, pick a channel and $p = 4$ points in time, concatenate, feed to a fully-connected layer).

K-beam minimax: efficient optimization for deep adversarial learning
J. Hamm and Y.K. Noh (2018)

When $f(\cdot, \cdot)$ is convex in its first argument and concave in the second, saddlepoints coincide with minimax points:

$$f(u^*, v^*) = \text{Max}_v \text{Min}_u f(u, v) = \text{Min}_u \text{Max}_v f(u, v).$$

Use alternating gradient descent, but track k candidate solutions, to account for the discontinuities and multiple solutions to the maximization problem.

Averaging weights leads to wider optima and better generalization
P. Izmailov et al.

A cyclical, linearly decreasing learning rate $\backslash \backslash \backslash \backslash$ yields an ensemble of solutions: one can average their forecasts or, since they tend to be dispersed on the edge of a low-cost region, their weights – wider optima generalize better (cf. entropy-SGD).

Cyclical learning rates for training neural networks
L.N. Smith (2015)

The *learning rate (LR) range test* estimates minimum and maximum values for the learning rate by computing the accuracy of the model for various learning rates (e.g., from 0 to 5 – you may go beyond 1) after just a few epochs.

The *cyclical learning rate* schedule linearly increases and decreases the learning rate between those bounds,

with 2 to 10 epochs per half-period: the learning rate will be close to the optimal value most of the time, but higher values will help escape saddle points (as in simulated annealing).

The idea is very similar to stochastic gradient descent with restarts, and related to adaptive learning rate methods (AdaDelta, AdaGrad, RMSProp, Adam, etc.).

Super-convergence: very fast training of neural networks using large learning rates
L.N. Smith and N. Taupin

The cyclical learning rate method can speed up training by up to an order of magnitude.

The optimal learning rate can be estimated as

$$\varepsilon^* = \varepsilon \frac{\theta_{i+1} - \theta_i}{2\theta_{i+1} - \theta_i - \theta_{i+2}}.$$

Constant step size stochastic gradient descent for probabilistic modeling
D. Babichev and F. Bach

Stochastic gradient descent with constant step size does not converge but, for exponential families, averaging the moment parameters (e.g., the probability $p \in [0, 1]$ for a Bernoulli distribution, *i.e.*, a logistic regression), rather than the natural parameters (the log-odds ratio) restores convergence.

Dropout is a special case of the stochastic delta rule: faster and more accurate deep learning
N. Frazier-Logue and S.J. Hanson

The *stochastic delta rule* (SDR) represents the weights of a neural net as Gaussian random variables $w_{ij} \sim N(\mu_{ij}, \sigma_{ij}^2)$, sampled on each forward iteration, whose parameters are updated. Dropout uses (fixed-parameter) Bernoulli distributions.

Measuring the intrinsic dimension of objective landscapes
C. Li et al. (2018)

Train your model on a random subspace of the space of weights (the model remains the same, but the weights are constrained to a linear subspace) and progressively increase the dimension: that at which the performance reaches 90% of the full network is the *intrinsic dimension*. It may depend on the network structure: for images, CNNs give a lower dimension.

Those random projections can also be used to compress networks.

Neural networks should be wide enough to learn disconnected decision regions
Q. Nguyen et al. (2018)

If no hidden layer (in a network with ReLU activations) is wider than the input, the decision regions are connected.

The lottery ticket hypothesis: finding small, trainable networks
K. Frankle and M. Carbin

A large network is an ensemble of smaller overlapping networks: some of those subnetworks have won the “initialization lottery” and, once pruned, can be trained – they work better than the initial, large network.

Harmonic analysis of neural networks
E.J. Candès (1996)

Continuous and discrete wavelets can be seen as (infinite and finite) 1-layer neural networks – they model functions as

$$f = \int \langle f, \psi_\gamma \rangle \psi_\gamma \mu(d\gamma)$$

$$\text{or } f = \sum \alpha_\gamma \psi_\gamma, \text{ where } \psi_\gamma(x) = \alpha^{-1/2} \psi\left(\frac{\langle u, x \rangle - b}{a}\right).$$

Manifold regularization with GANs for semisupervised learning
B. Lecouat et al. (2018)

GANs can be leveraged for semi-supervised learning: have the discriminator determine the class of the input or whether it was generated. (Self-training is also sometimes used: label the unlabeled data with a classifier trained on the labeled data, then re-train on the expanded dataset.)

Manifold regularization ensures the classifier does not change as we move away from the data (orthogonally), with a penalty of the form

$$\int \|\nabla_{\mathcal{M}} f\| dp \approx \frac{1}{n} \sum_i \nabla_z f(g(z^{(i)}))$$

$z^{(i)}$: latent generator variables

g : generator

f : discriminator, classifier

\mathcal{M} : data manifold

p : distribution of the data on \mathcal{M} .

The relativistic discriminator: a key element missing from standard GAN
A. Jolicoeur-Martineau

The generator and discriminator of a GAN minimize their loss functions

$$\text{loss}_D = \mathbb{E}_{x \sim \text{Data}} [f_1(D(x))] + \mathbb{E}_{z \sim \text{noise}} [f_2(DGz)]$$

$$\text{loss}_G = \mathbb{E}_{x \sim \text{Data}} [g_1(D(x))] + \mathbb{E}_{z \sim \text{noise}} [g_2(DGz)],$$

often with $g_1 = f_1$, $g_2 = -f_2$, *i.e.*, the loss is the same, up to the sign (saturating GAN) or with $g_1 = f_2$,

$g_2 = f_1$, i.e., the loss is the same, but the real and simulated data have been swapped (non-saturating GAN).

The “relativistic GAN” uses

$$\begin{aligned}\text{loss}_D &= - \mathbb{E}_{\substack{x \sim \text{Data} \\ z \sim \text{Noise}}} [\log \sigma(Dx - DGz)] \\ \text{loss}_D &= - \mathbb{E}_{\substack{x \sim \text{Data} \\ z \sim \text{Noise}}} [\log \sigma(DGz - Dx)]\end{aligned}$$

to ensure that the probability of real data decreases as that of the fake data increases.

χ^2 generative adversarial network C. Tao et al. (2018)

GANs can be trained with:

- f -divergences

$$P(p||q) = \int f \left(\frac{p(x)}{q(x)} \right) p(x) dx$$

where p is the data and q the generator – but this is numerically unstable;

- Integral probability metrics (IPM), such as Wasserstein 1 (earth mover’s distance)

$$\sup_{\|D\| \leq 1} \mathbb{E}_{X \sim \text{Data}} D(X) - \mathbb{E}_{X \sim q} D(X)$$

- RKHS distances,

$$\text{MMD}(p, q) = \left\| \mathbb{E}_{X \sim p} \kappa(\cdot, X) - \mathbb{E}_{X \sim q} \kappa(\cdot, X) \right\|_H.$$

The χ^2 GAN is a special case of all three.

First order generative adversarial networks C. Seward et al.

An *adversarial divergence* between probability distributions p, q on X is a function of the form

$$\tau(p||q) = \sup_{g \in \mathcal{G}} E_{p \otimes q}[g]$$

for some $\mathcal{G} \subset \mathcal{C}^0(X \times X)$.

A *critic-based adversarial divergence* is of the form

$$\tau(p||q) = \sup_{f \in \mathcal{F}} E_{p \otimes q}[m_f - r_f]$$

where $\mathcal{F} \subset \mathcal{C}^0(X)$ (e.g., 1-Lipschitz), $m_f(x, y) = m_1(f(x)) - m_2(f(y))$, $m_1, m_2 : \mathbf{R} \rightarrow \mathbf{R}$ and $r : \mathcal{F} \rightarrow \mathcal{C}^0(X \times X)$.

The Wasserstein divergence between Dirac masses

$$\tau(\delta_a||\delta_b) = \sup_{f \in \mathcal{F}} f(a) - f(b) = |a - b|$$

can be written

$$4\tau(\delta_a||\delta_b) = \sup_{f \in \mathcal{F}} f(a) - f(b) - |a - b| \left(\frac{f(a) - f(b)}{a - b} \right)^2;$$

this motivates the *penalized Wasserstein divergence*,

$$\tau(p||q) = \sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{y \sim q} f(y) - \lambda \mathbb{E}_{\substack{x \sim p \\ y \sim q}} \frac{(f(x) - f(y))^2}{\|x - y\|}$$

(one can also add another penalty, involving ∇f).

Evolutionary generative adversarial networks C. Wang et al.

Evolve a population of generators to adapt to the environment (discriminator), mutating them by running a few gradient descent steps for the following loss functions

$$\begin{aligned}\text{loss}(G) &= \mathbb{E}_{z \sim \text{noise}} \log(1 - DGz) \\ \text{loss}(G) &= \mathbb{E}_{z \sim \text{noise}} -\log(DGz) \\ \text{loss}(G) &= \mathbb{E}_{z \sim \text{noise}} (DGz - 1)^2\end{aligned}$$

where the first (second) minimizes (maximizes) the log-probability of the discriminator being correct (mistaken).

To promote diversity and avoid mode collapse, note that the discriminator labels collapsed points as fake, with obvious counter-measure, i.e., big gradients, and use those gradients as a diversity measure.

Large scale GAN training for high-fidelity natural image synthesis A. Brock et al.

To train large-scale GANs:

- Increase the batch size;
- Train with Gaussian noise but sample using a truncated Gaussian;
- Use orthogonal regularization, $\|W'W - I\|_F^2$, without the diagonal $\|W'W \odot (1 - I)\|_F^2$.

Synthesizing programs for images using reinforced adversarial learning Y. Ganin et al.

GAN to generate programs (sequences of graphical primitives) to generate images, instead of raw pixels (deep reinforced adversarial learning).

Globally and locally consistent image completion S. Iizuka et al. (2017)

Use a GAN for image completion, with both a global and a local discriminator.

Learning to see in the dark C. Chen et al.

Instead of using separately learned algorithms for denoising, deblurring, etc., learn an end-to-end image processing pipeline.

**DeepMasterPrints: generating MasterPrints
for dictionary attacks
via latent variable evolution**
P. Bontrafer et al.

Use a GAN to generate fingerprints, then explore the latent space with CMA-ES to find a master print.

Small fingerprint sensors (on mobile phones), which only image part of the finger, are very vulnerable to that attack.

Learning deep generative models of graphs
Y. Li et al. (2018)

To sequentially generate random graphs (molecules, parse trees, graphical model structures), use a GRU graphnet to decide, at each step, where to add an edge or a node.

**Meta-learning for semi-supervised
few-shot classification**
M. Ren et al. (2018)

A **prototypical network**, for a classification problem, learns an embedding, defines prototypes as the averages of the embeddings of the observations in each class, and uses the squared distances to the prototypes in a softmax, $p \propto \exp(-d^2)$.

For semi-supervised learning, compute the prototypes from the labeled data, estimate the class of the unlabeled items (soft- k -means, with a “distraction cluster” for potential unknown labels, or soft-masking of far-away observation) and refine the prototypes.

Learning to generate classifiers
N. Guttenberg and R. Kanai

Meta-learning (*i.e.*, learning a mapping from datasets of predictors and labels to classifiers) with attention outperforms SVM, random forests, k -NN, or xgboost.

**Fastfood – approximating kernel expansions
in loglinear time**
Q. Le et al.

The decision function, in an SVM, is of the form

$$f(x) = \sum_1^N \alpha_i k(x_i, x),$$

where N is the number of observations and the nonzero α_i correspond to the support vectors. We usually prefer to compute the kernel directly, without explicitly computing the embedding, $k(x, y) = \langle \phi(x), \phi(y) \rangle$, but, for large datasets, a random embedding (*random kitchen sink*) may be preferable:

$$Z_{ij} \sim N(0, \sigma^{-2})$$

$$\phi_j(x) = \frac{1}{\sqrt{n}} \exp[i(Zx)_j]$$

is an approximation of a RBF (radial basis function) kernel. Instead of a fully Gaussian kernel, consider

$$Z = \frac{1}{\sigma\sqrt{d}} SHG\Pi HB$$

where $H = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}^{\otimes d}$, B is a random diagonal matrix with entries ± 1 , G is a random diagonal Gaussian, and S is diagonal.

Using matrices to model symbolic relationships
I. Sutskever and G. Hinton

A **linear relational embedding** (LRE) learns vector and matrix representations of objects and relations such that $x\mathcal{R}y$ corresponds to a matrix product $\mathcal{R}x = y$; a *matrix relational embedding* uses matrices for both objects and relation: the relation matrices are smaller, and one can also consider higher-order relations (relations between relations).

**Monotonic calibrated interpolated
look-up tables**
M. Gupta et al. (2016)

Lattice regression approximates a function by interpolating it on a grid – the values on the vertices are chosen to minimize the loss, *after interpolation*, at the points already observed. Multilinear interpolation uses 2^D points, e.g.,

$$f(x, y) = (1-x)(1-y)\theta_{00} + x(1-y)\theta_{10} + (1-x)y\theta_{01} + xy\theta_{11}$$

(for $D = 2$), where the weights $x^\delta(1-x)^{1-\delta}y^\varepsilon(1-y)^{1-\varepsilon}$, $\delta, \varepsilon \in \{0, 1\}$, are known, and the model is linear in its parameters θ . Monotonicity is easy to check. One could use any other way of writing (x, y) as a barycenter of $(0, 0)$, $(1, 0)$, $(0, 1)$, $(1, 1)$. **Simplex interpolation** uses only $D + 1$ vertices, corresponding to a partition of the hypercube along the hyperplanes $x_k = x_\ell$. The simplex containing a point can be obtained by sorting its coordinates (in $O(D \log D)$ time). The model is not rotationally invariant: all the simplices contain $(0 \cdots 0)$ and $(1 \cdots 1)$; for a better fit, ensure all the constraints go in the same direction, e.g., everything increasing.

Add penalties such as

$$\sum \left(\frac{\partial f}{\partial x_i} \right)^2, \quad \sum \left(\frac{\partial^2 f}{\partial x_i^2} \right)^2, \quad \text{or} \quad \sum \left(\frac{\partial^2 f}{\partial x_i \partial x_j} \right)^2$$

to make the function flatter or more linear.

Add coordinatewise, piecewise (but still monotonic) transformations, with k knots, at equally spaced quantiles.

To deal with missing values, replace the lattice $\{0, 1\}^D$ with $\{0, 1, \text{NA}\}^D$ and interpolate on $([0, 1] \sqcup \{\text{NA}\})^D$.

There are many constraints and penalties: try sampling from them at each iteration to speed up the computations.

Lattice regression
E.K. Garcia and M.R. Gupta

***Fast and flexible monotonic functions
with ensembles of lattices***
K. Canini et al. (2016)

Build an ensemble of (monotonic) lattice regression models, each using a small number of variables:

- Train lattices on all pairs of features and compute their “torsion”, $[(x_{11} - x_{10}) - (x_{01} - x_{00})]^2$, a measure of nonlinear interaction;
- Assign the features to the different models, with each feature used at least once, and the feature count otherwise proportional to the median torsion, and maximize the weighted torsion of the ensemble (where the weight decreases as a pair is repeated).

***Deep gradient compression:
reducing the communication bandwidth
for distributed training***
Y. Lin et al. (2018)

Most of the gradient exchanges in distributed stochastic gradient descent are not needed:

- Only send large gradients;
- Accumulate them locally, and clip them, before sending;
- Do not use momentum for old (unsent) gradients – it would be stale;
- During training, use little sparsification and a low training rate.

***meProp: sparsified backpropagation
for accelerated deep learning
with reduced overfitting***
X. Sun et al. (2017)

Sparsified backpropagation, *i.e.*, only backpropagating the k largest gradients (1% to 4%) does not slow down training.

Deep image prior
D. Ulyanov et al.

Use untrained networks (reservoir computing, random projections).

***Secure ML: a system for scalable
privacy-preserving machine learning***
P. Mohassel and Y. Zhang

Machine learning models can be trained even if the data is (and should remain) on separate servers (e.g., financial and medical data).

***Trainable calibration measures for neural
networks from kernel mean embeddings***
A. Kumar et al. (2018)

Neural networks for classification problems are not calibrated: the scores cannot be interpreted as probabilities – we would like $P[\text{correct}] = \text{Max}_k p_k(x)$. The calibration error could be added to the loss function, but it is discontinuous. Instead, use

$$\text{MMCE} = \sum_{ij} \frac{(c_i - s_i)(c_j - s_j)\kappa(s_i, s_j)}{N^2}$$

where $c_i = \mathbf{1}_{\text{correct}} = \mathbf{1}_{y_i = \text{Argmax}_k p_k(x_i)}$ and $s_i = \text{Max}_k p_k(x_i)$.

Machine learning for trading
G. Ritter (2017)

Reinforcement learning on simulated data can help devise trading strategies with arbitrary transaction cost models (e.g., given by an algorithmic blackbox).

Algorithms for inverse reinforcement learning
A.Y. Ng and S. Russell

The reward function of a known, finite-state MDP can be recovered from its (deterministic) optimal policy, via a linear program; to avoid degenerate solutions, e.g., $R \equiv 0$, maximize the difference between the optimal policy and the others,

$$\sum_{s \in S} Q(s, \pi(s)) - \text{Max}_{a \neq \pi(s)} Q(s, a).$$

For large state spaces, approximate the reward function as a linear combination of basis functions, $R(s) = \sum \alpha_i \phi_i(s)$, ϕ_i fixed. This approach is still applicable if the optimal policy is only known through sample trajectories (but requires iteratively solving linear programs, to progressively improve the fit).

***Solving the Rubik’s cube
without human knowledge***
S. McAleer et al.

Only one state, the final state, has a reward: start there and estimate the value and optimal policy for states farther and farther away from it.

***Independent interpretable lasso:
a new regularizer for sparse regression
with uncorrelated variables***
M. Takada et al. (2018)

To avoid selecting correlated variables, ease interpretation, and recover correct signs, use a penalty of the form $\lambda \|\beta\|_1 + \mu |\beta|' R |\beta|$, where $R_{ij} \geq 0$ measures the similarity between i and j .



$$\lambda = 1, \mu = \frac{1}{2}, R = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

***High-dimensional regression in practice:
an empirical study of finite-sample prediction,
variable selection and ranking***
F. Wang et al. (2018)

Prefer the lasso. Other penalized regression, variable selection and variable ranking algorithms include the elastic net, ridge regression, SADC (an ℓ^1 penalty for small values, ∇), the Dantzig selector

$$\underset{\beta: \|X'(Y-X\beta)\|_\infty \leq \lambda}{\text{Argmin}} \quad \|\beta\|_1,$$

stability selection (run a variable selection on resampled datasets and keep the variables selected at least $\tau\%$ of the time).

R implementations in `glmnet`, `ncvreg` (SADC), `flare` (Dantzig), `c060` (stability selection).

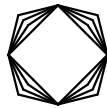
***Identifying groups
of strongly correlated variables
through smoothed ordered weighted ℓ_1 norms***
R. Sankaran et al. (2017)

The ordered weighted L^1 norm (OWL) is

$$\Omega(w) = \sum c_i |w|_{(i)}$$

where $|w|_{(i)}$ is the i th largest absolute value of the coordinates of w and c_i is a fixed non-increasing sequence (e.g., an arithmetic sequence: OSCAR).

As a penalty, it has a sparsifying effect, selecting groups of variables with the same coefficients



It is the Lovász extension of the cardinality-based modular function $P(A) = f(|A|)$, where $f(i) = c_1 + \dots + c_i$.

The ℓ^2 relaxation is similar, but does not force the coefficients to be exactly equal.

***Finite-time analysis
of the multiarmed bandit problem***
P. Auer et al. (2002)

For the multiarm bandit problem, UCB_1 , which plays the machine j maximizing

$$x_j + \sqrt{\frac{2 \log n}{n_j}},$$

UCB_2 (similar, with a division in epochs), and ε -greedy, with $\varepsilon \propto 1/n$, achieve logarithmic regret.

***Multi-fidelity blackbox optimization
with hierarchical partitions***
R. Sen et al. (2018)

UCB-style multi-arm bandit algorithms can be applied to black-box optimization of expensive functions for

which cheap, coarse approximations (early stopping) are available, by hierarchically partitioning the domain.

Contextual memory trees
W. Sun et al.

Store key-value pairs in the leaves of a (balanced, binary) tree, with a classifier at each node; at query time, update the classifiers to improve retrieval.

The case for learned index structures
T. Kraska et al.

Many data structures can be replaced by learned models, leveraging GPUs/TPUs.

B-trees (range indices, in databases) map a key to a position in a sorted array, *i.e.*, estimate a cumulative distribution function (CDF); they can be replaced with a neural net. The maximum error measured on the whole dataset defined the size of the window to examine (the B-tree is actually similar: it retrieves a whole “page”, instead of the single value it actually needs). It is possible to replace only part of the data structure with a neural net, or arrange several neural nets in a tree.

Hash functions can also be replaced by learning the CDF of the key, and dilating it to get a hash map.

A good hash function for a Bloom filter would have many key/key and non-key/non-key collisions, but few key/non-key collisions. Bloom filters can be replaced by a key/non-key classifier, with a smaller Bloom filter for the set of false negatives, to keep the false negative rate to zero.

***Clustering by passing messages
between data points***
B.J. Frey and D. Dueck

Use exemplars:

$$\begin{aligned} s_{ij} &: \text{similarity} \\ r_{ik} &\leftarrow s_{ik} - \text{Max}_{k' \neq k} a_{ik'} + s_{ik'} \\ a_{ik} &\leftarrow \text{Min} \left\{ 0, r_{kk} + \sum_{i' \neq i, k} (r_{ik'})_+ \right\} \\ a_{kk} &\leftarrow \sum_{j' \neq k} (r_{ik'})_+ \\ \text{exemplar}(i) &= \text{Argmax}_k a_{ik} + r_{ik} \end{aligned}$$

with damping (*i.e.*, updates of the form $x \leftarrow (1 - \lambda)x + \lambda x_{\text{new}}$) and only exchanging messages between nearby or similar points.

***Distributional regression forests
for probabilistic precipitation forecasting
in complex terrain***
L. Schlosser et al.

Linear regression models the mean of the response variable as a function of the predictors.

Non-homogeneous Gaussian regression (NGR) models both mean and variance.

GAMLSS uses a GAM for location, scale and shape (`gamlss`, `gamboostLSS`).

To account for interactions and non-smooth dependencies, one can use a random forest of regression trees with GAMLSS leaves (`disttree`).

***Statistical detection
of systematic election irregularities***
P. Klimek et al. (2012)

Ballot stuffing (only one type of election irregularity) can be detected by looking at:

- The winner votes vs turnout density plot, which should not show an abnormal high-turnout, high rate cluster;
- The cumulative number of votes as a function of turnout, which should end with a plateau.

***A fast and objective multidimensional
kernel density estimation method: fastKDE***
T.A. O'Brien et al. (2016)

The choice of kernel and bandwidth, for kernel density estimation (KDE), is often arbitrary. Use κ , where

$$\tilde{\kappa}(t) = \frac{n}{2(n-1)} \left[1 + \sqrt{1 - \frac{4(n-1)}{n^2 |\mathcal{C}(t)|^2} I(t)} \right]$$

$$\mathcal{C}(t) = \frac{1}{n} \sum e^{ix_j \cdot t} \text{ empirical characteristic function}$$

$$\tilde{\kappa} = \mathcal{F}^{-1} \kappa \text{ inverse Fourier transform}$$

and I filters out frequencies t with $|\mathcal{C}(t)| \leq 2\sqrt{n-1}/n = \mathcal{C}_{\min}$ and possibly a few more (e.g., outside $[-t^*, t^*]$, where for half the t in $[-t^*, t^*]$, $\mathcal{C}(t) \geq \mathcal{C}_{\min}$). The optimal kernel can be computed with the nuFFT (non-uniform FFT).

This can be generalized to multivariate KDE.

***Controllable conformal maps
for shape deformation and interpolation***
O. Weber and C. Gotsman (2010)

To deform a (simply-connected) shape Ω , use a conformal transformation f . It suffices to specify the *angular factor* $\theta = (\log f')|_{\partial\Omega}$. the conformal transformation can be recovered as follows:

- Extend θ harmonically on Ω ;
- Compute its harmonic conjugate ϕ (Hilbert transform);
- Consider the holomorphic function $g = \phi + i\theta$;

- Solve $f' = \exp g$ – since f' does not vanish, f is conformal.

Generalized Cauchy coordinates use the Cauchy formula, a polynomial discretization of $\partial\Omega$, a quadratic approximation of f on each edge, with two values at each corner.

An invitation to noncommutative algebra
C. Walton

Noncommutative algebra studies *representations* of noncommutative algebras, e.g., the quaternions, or the *Weyl algebra*

$$\frac{k\langle x, y \rangle}{(yx - xy - 1)}$$

(which models differential operators, with $y = d/dx$, and does not have finite-dimensional representations – to see it, take the trace), *deformations* of commutative structures, e.g.,

$$\mathbf{C}_q[x, y] = \frac{k\langle x, y \rangle}{yx - qxy}, \quad q \in \mathbf{C}^\times,$$

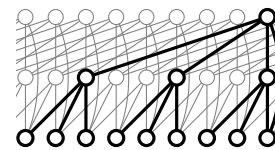
and deformations of group actions (into Hopf algebra actions), e.g.,

$$H_q = \frac{\mathbf{C}\langle g, g^{-1}, h \rangle}{(gg^{-1} - 1, g^{-1}g - 1, gh - q^2hg)}$$

(as $q \rightarrow 1$, $g, g^{-1} \rightarrow 1$, $h \rightarrow \partial_y$).

***An empirical evaluation of
generic convolutional and recurrent networks
for sequence modeling***
S. Bai et al. (2018)

Use *temporal convolutional networks* (TCN) to model sequences: no pooling, causal dilated convolutions and residual connections.



***Detecting malicious PowerShell commands
using deep neural networks***
D. Hendler et al.

To detect malicious commands in the logs, use an ensemble of:

- 3-grams tf-idf logistic regression;
- bag-of-words tf logistic regression;
- (CNN,MaxPool)²CNN⁴MaxPool(FC,Dropout)⁴FC;
- CNN, MaxPool, (FC, Dropout)²,

on a dataset of 100,000 commands, 10% malicious.

BERT: pre-training of deep bidirectional transformers for language understanding
J. Devlin et al.

Most language models only rely on the left context, *i.e.*, model $P[w_{n+1}|w_{1:n}]$; BERT uses *transformers* (*i.e.*, attention-based networks) conditioned on both the left and right context, $P[w_n|w_{1:n-1}, w_{n+1:N}]$. ELMo is similar, but uses separate left-to-right and right-to-left LSTMs, which are only concatenated at the end.

Are distributional representations ready for the real world? Evaluating word vectors for grounded perceptual meaning
L. Lucy and J. Gauthier

Word embeddings cannot recover *semantic norms*, *i.e.*, entailments such as *alligator* \rightarrow *is green, animal, reptile, eats people, has teeth, etc.*

Morph-fitting: fine-tuning word vector spaces with simple language-specific rules
I. Vulić et al.

To modify a trained word embedding to account for known synonym and antonym pairs (e.g., different word endings in morphologically rich languages), use the following loss function

$$\begin{aligned} & \sum_{x_\ell \sim x_r} \text{ReLU}(\delta + x_\ell t_\ell - x_\ell x_r) + \text{ReLU}(\delta + x_r t_r - x_\ell x_r) \\ & + \sum_{x_\ell \not\sim x_r} \text{ReLU}(\delta + x_\ell t_r - x_\ell x_\ell) + \text{ReLU}(\delta + x_\ell t_r - x_r x_r) \\ & + \lambda \sum_x \|x^{\text{init}} - x\|_2^2 \end{aligned}$$

where t_i is the word closest to x_i (in the current mini-batch).

BLEU: a method for automatic evaluation of machine translation
K. Papineni et al.

TextRank: bringing order into texts
R. Mihalcea and P. Tarau

To extract keywords from a text, apply undirected PageRank to the cooccurrence graph (vary the window size) between words or sequences of words (try different POS, e.g., nouns and adjectives).

For sentence extraction (summarization), use sentences as nodes and similarity (number of common words, divided by the log of the lengths) as weights.

Multimodal word distributions
B. Athiwaratkun and A.G. Wilson

Extend the skipgram model by replacing the point representation of words with Gaussian mixtures, allowing for both polysemy and entailment (e.g., *jazz* \subset *music*).

Enriching word vectors with subword information
P. Bojanowski et al.

FastText fits a skipgram model to n -grams and represents words as the sum of their n -grams – it is aware of word morphology and can deal with unknown words.

A simple but tough-to-beat baseline for sentence embeddings
S. Arora et al. (2017)

Turning word embeddings into sentence embeddings by naively averaging the embeddings of the words in the sentence does not work that well; try the following simple modification:

- Use a weighted average, $w = \frac{a}{a + P(\text{word})}$;
- Remove the first principal component (it contains stopwords and other non-informative words).

In the word2vec (CBOW) model

$$P[w|w_1 \cdots w_5] \propto \exp\left(v_w \cdot \frac{1}{5} \sum_i v_{w_i}\right)$$

frequent words are under-sampled when computing the gradient, to speed up training and give more regular embeddings.

Supervised learning of universal sentence representations from natural language inference data
A. Conneau et al. (2018)

InferSent uses a supervised task, *natural language inference* (recognizing if a sentence entails, contradicts or is unrelated to another) with a bidirectional LSTM RNN with max pooling to compute sentence embeddings.

Skip-thought vectors
R. Kiros et al.

To compute sentence embeddings, forecast a sentence from nearby sentences (as in word2vec's CBOW) using a GRU RNN trained on the BookCorpus dataset (unpublished authors).

For out-of-vocabulary words, learn a mapping from the word2vec space (using unpenalized least squares).

Vader: a parsimonious rule-based model for sentiment analysis of social media text
C.J. Hutto and E. Gilbert (2014)

Vader is an MIT-licensed sentiment lexicon, targeted at short, Twitter-like sentences, augmented with a few simple rules for all-caps, punctuation (exclamation marks), degree modifiers (extremely, marginally, etc.), contrastive conjunctions (but) and negations, available in NLTK.

Deep contextualized word representations
C. Clark et al. (2018)

ELMo uses, as sentence embedding, all the internal layers (not just the first and the last ones) of a deep bidirectional LSTM network trained as a language model (*i.e.*, to predict the next/previous/missing word).

Universal sentence encoder
C. Cer et al. (2018)

Train a model, e.g., a **transformer** (attention-based network, instead of an LSTM) or a **deep averaging network** (DAN: the embeddings of the inputs words and bigrams are averaged and fed to a deep neural network – the word embeddings are trained so that their averages be useful as sentence embeddings) on several tasks (skip-thought, conversation input-response, classification).

Pretrained models are available on TFHub.

Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints
N. Mrkšić et al. (2017)

Attract-Repel modifies a word embedding to account for synonyms and antonyms (some come from cross-lingual sources, which can also provide multilingual embeddings) using a margin loss on positive (actual synonyms or antonyms) or negative pairs.

Morph-fitting: fine-tuning word vector spaces with simple language-specific rules
I. Vulić et al. (2017)

Modify a vector embedding to account for synonyms (inflections, e.g., V, V-ed, V-ing, in English) and antonyms (derivations, e.g., negative English prefixes: dis, un, in, im, il, ir, mis, non, anti) with margin loss.

Concatenated power mean word embeddings as universal cross-lingual sentence representations
A. Rückle et al. (2018)

To compute a sentence embedding, concatenate several “averages” (arithmetic, maximum, minimum) of several pretrained word embeddings (GloVe, word2vec, Attract-Repel, MorphSpecialized).

A neural probabilistic language model
J. Bengio et al. (2003)

The first layer of an n -gram neural language model (predicting the next word from the previous n) can be used as a word embedding.

Deep-learning-based cryptocurrency sentiment construction
S. Nasekin and C.Y.S. Chen (2018)

Word2vec and RNN-based sentiment index, trained with StockTwits data – users often label their messages with “bullish” or “bearish”.

Modeling snow crystal growth I
J. Gravner and D. Griffeath (2006)

Packard’s digital snowflake model is a cellular automaton on a hexagonal lattice in which a site with one (or at least one, or between 1 and 3, etc.) occupied neighbour becomes occupied.

A deep learning framework for financial time series using stacked autoencoders and long-short-term memory
W. Bao et al. (2016)

Try to forecast six equity indices from prices, volumes, technical indicators and macro variables, denoised with a wavelet transform [doesn’t this introduce some look-ahead bias?], dimension-reduced with an auto-encoder, and fed to an LSTM.

Optimal timing and tilting of equity factors
H. Dichtl et al. (2018)

To time factors,

- Take
 - Some (US) macroeconomic indicators;
 - Some technical indicators of those factors;
 - Some cross-sectional factor characteristics, e.g., earning yields of the long-short quintile portfolios, momentum, volatility [these are technical indicators], centrality in the (weighted) complete correlation graph [they use the MST], distance to market [just a beta];
- Perform some dimension reduction (only keep the first principal component) in each group;
- Find the solution of the mean-variance optimization problem of the form $w = \theta z'$ where w are the (unknown) portfolio weights, z are the principal components (and an intercept) and θ are the new unknowns (Brandt–Clara parametric portfolio policy).

Big data and AI strategies: ML and alternative data approach to investing
M. Kolanovic (JPM, 2017)

The big data revolution is made possible by the combination of data, algorithms and hardware. Alternative data can be classified from their origin: humans, business processes or machines (sensors). Beware of seasonality, sampling bias and short history. Examples include sentiment (for instance, stocks, regions, sectors, indices), Twitter, news, Wikipedia, company websites, web traffic (Alexa), search (Google trends), job postings (Glassdoor, LinkedIn), purchase receipts.

- Twitter sentiment, for 100 stocks in the S&P 500, aggregated every minute and smoothed with a 10-day exponential moving average, to forecast the S&P 500 returns for the next 2 days, with time-changing betas (Kalman), for a long-short strategy;

- Ravenpack news data, average daily sentiment score (ESS) for currencies, commodities, countries, where relevance > 75, for a daily long-short strategy;
- Email receipts, for 30 companies in the S&P 500: weekly change of dollar spent, number of orders, number of buyers;
- Mobile phone location, assuming sales are proportional to traffic, to predict if a company will beat expectations;
- Satellite images to monitor retail traffic, real estate traffic, metal production and storage, factory employment (Bollinger bands on quarterly traffic);
- Penalized regression to forecast S&P, 10y, DXY, gold returns (daily), from their 1m, 3m, 6m, 12m lagged values, on a 2-year window;
- k -NN to forecast risk premia from macro indicators
- Kalman filter (time-varying beta) for pairs trading;
- Xgboost to forecast daily sector returns from macro factors;
- Logistic regression to forecast if call overwriting will outperform a long-only strategy, from 10 investment factors;
- SVM to forecast if the P&L of a rolling long 1m ATM EURUSD will be > 20bp, or < -20bp, from the first 50 to 200 principal components of 400 predictors (volatility, skew, spot, basis, interest rate, equity/commodity/bond indices, economic activity, IMM position; their levels and 1w, 1m changes);
- Random forest to forecast 1-month returns from investment factors;
- 2-class hidden Markov model (HMM) to identify up and down states (low and high volatility): buy the S&P 500 in up markets, keep cash in down markets;
- Hierarchical risk parity;
- PCA on daily changes in the USDJPY implied volatility to help describe its variations;
- Statistical risk models;
- Multilayer perceptron to forecast next day sector returns from 8 macro factors (just 8 predictors: oil, gold, dollar, bonds, economic surprise, 10y-2y spread, IG and HY credit spreads);
- LSTM to forecast monthly returns from the past 3 years (does not work);
- SVM to forecast daily FX returns from 20 features, obtained by dimension reduction with a restricted Boltzman machine (RBM) using 10 days of daily returns for 10 commodities (100 predictors);
- Reinforcement learning (combined with supervised learning to forecast returns), for trading.

The report also includes R and Python sample code (web scraping and simple models), a long list of data providers, and 16 pages of references.

***Essentially no barriers
in neural network energy landscape***
F. Draxler et al. (2018)

Neural networks have many local minima. Linear interpolation between them suggests they are separated by high-loss barriers, but the AutoNEB algorithm can find paths of almost constant loss.

The *elastic band* model finds points minimizing $\sum \text{loss}(p_i) + \frac{1}{2} \sum k \|p_{i+1} - p_i\|^2$ (where k is the spring stiffness). In the *nudged elastic band* model, the loss force only acts perpendicularly to the path, and the spring force parallelly, $g = \nabla \text{loss}|_{\perp} + \nabla \text{spring}|_{\parallel}$, so that the spring force redistribute the points on the path without straightening it. Instead of choosing the spring stiffness k , one can remove the springs altogether and simply redistribute the points on the path at each iteration. The *AutoNEB* algorithm progressively increases the number of points.

While the algorithm finds *local* minimum energy paths, the loss may increase too much: this may be remedied by considering a graph of low-energy paths between several local minima.

***Quickshift++: provably good initializations
for sample-based mean shift***
H. Jiang et al. (2018)

The **MeanShift** clustering algorithm is a gradient ascent on a kernel density estimator f .

The *QuickShift* algorithm is a discretized variant, moving each sample to the closest one with a higher density in a radius τ ball.

The *QuickShift++* algorithm is a variant stopping when it reaches a cluster core, rather than a local mode.

A *cluster core* M of a topological space X is a connected component of

$$\left\{ x \in X : f(x) \geq (1 - \beta) \max_{y \in M} f(y) \right\}.$$

The k -NN density estimator in \mathbf{R}^d is $f(x) \propto d_k(x)^{-d}$, where $d_k(x)$ is the distance to the k th nearest neighbour.

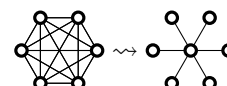
***Representation tradeoffs
for hyperbolic embeddings***
C. De Sa et al. (2018)

To embed a tree in the Poincaré ball (Sarka's construction), start by placing the root a at the origin,

- Apply a reflection, if needed, to put a at the origin;
- Add a 's children on a radius τ circle centered on a , equally-spaced, but as far away from a 's parent as possible;
- Apply the reflexion to go back to the original position (we still have a circle and equal angles, but the circle is no longer centered on a);
- Iterate with a 's children.

In higher dimensions, use *spherical encoding* to put the points on the sphere (e.g., using vertices of a hypercube).

For a general graph, add *Steiner nodes*.



Reconstructing approximate tree metrics
I. Abraham et al. (2007)

Instead of embedding your data (e.g., a graph) in a Euclidean space, try embedding it in a tree.

To embed a graph in a tree:

- Choose a “root” node r ;
- Find nodes p, q maximizing the *Gromov product*

$$(p|a)_r = \frac{1}{2}(p(p, q) + d(q, r) - d(p, q))$$

and add a **Steiner node** $\triangle \rightsquigarrow \circ$ at distances $(q|r)_p, (p|r)_q, (p|q)_r$ of p, q, r ;

- Remove q and iterate.

Learning continuous hierarchies in the Lorentz model of hyperbolic geometry
M. Nicket and D. Kiela (2018)

Graphs, especially trees, are more naturally embedded in hyperbolic spaces, e.g., the Poincaré disk, or the *hyperboloid model*.

Anonymous walk embeddings
S. Ivanov and E. Burnaev (2018)

An **anonymous walk** on a graph is a random walk (of length ℓ) with the nodes replaced with numbers (for instance, both ABCBC and CDBDB map to 12323). They give a sparse embedding and, with word2vec (anonymous walks are words, sets of walks starting at the same node sentences, graphs documents), a dense one.

Bayesian optimization of combinatorial structures
R. Baptista and M. Poloczek (2018)

To maximize a function f on $\{0, 1\}^d$, model it using only interactions of order up to k , e.g.,

$$f(x) = \alpha_0 + \sum_i \alpha_i x_i + \sum_{i < j} \alpha_{ij} x_i x_j \quad (k = 2),$$

estimate α from the points already evaluated with a sparse Bayesian regression (and a Horseshoe prior), sample a value for α , and approximately maximize f_α (with an L^1 or L^2 penalty) with a SDP relaxation.

Accurate uncertainties for deep learning using calibrated regression
V. Kuleshov et al. (2018)

Bayesian uncertainties are often inaccurate (variational inference is an approximation): to recalibrate a Bayesian deep learning model, *i.e.*, a model $x \mapsto F_x = \text{cdf of } y$, learn an auxiliary model (isotonic regression) $R : [0, 1] \rightarrow [0, 1]$ such that $R \circ F_x$ be calibrated.

Large-scale sparse inverse covariance estimation via thresholding and max-det matrix completion
R.Y. Zhang et al. (2018)

The *graphical lasso*

$$\underset{X \succ 0}{\text{Minimize}} \text{tr} CX - \log \det X + \lambda \sum_{ij} |X_{ij}|$$

to find a sparse concentration matrix X from a sample covariance matrix C can be approximated with a simple heuristic, first soft-thresholding ~~the~~ the covariance $C \mapsto C_\lambda$, then solving the *maximum determinant matrix completion* (MDMC) problem.

$$\begin{aligned} \text{Find} \quad & X \succ 0 \\ \text{To minimize} \quad & \text{tr} C_\lambda X - \log \det X \\ \text{Subject to} \quad & X_{ij} = 0 \text{ if } (C_\lambda)_{ij} = 0, \text{ i.e., if } |C_{ij}| \leq \lambda \end{aligned}$$

(There is a recursive closed form solution if the graph is chordal but, in practice, that rarely happens unless it is a tree.)

Conditional neural processes
M. Garnelo et al. (2018)

Neural nets can replace Gaussian processes to model functions $y = f(x)$:

$$\begin{aligned} h_\theta : \begin{cases} \mathbf{R}^N \times \mathbf{R} \rightarrow \mathbf{R}^d \\ (x, y) \mapsto r \end{cases} & a : \begin{cases} \mathbf{R}^d \times \dots \times \mathbf{R}^d \rightarrow \mathbf{R}^d \\ (r_1, \dots, r_d) \mapsto r \end{cases} \\ g_\theta : \begin{cases} \mathbf{R}^N \times \mathbf{R}^d \rightarrow \mathbf{R}^p \\ (x, r) \mapsto \phi \end{cases} \end{aligned}$$

where h_θ and g_θ are neural nets, a is some aggregation function, r summarizes the information so far, ϕ are the parameters of the distribution of y , e.g., $\phi = (\mu, \sigma^2)$ for a Gaussian.

Optimization, fast and slow: optimally switching between local and Bayesian optimization
M. McLeod et al. (2018)

- Use Bayesian optimization, with *predictive entropy search* (PES: maximize the expected change in information content), to find a convex region likely to contain the global minimum;
- Use Bayesian optimization, with a different acquisition function, to ensure the region indeed contains a global minimum;
- Find the minimum, using local search (L-BFGS).

Adaptive three operator splitting
F. Pedregosa and G. Gidel (2018)

To minimize a function of the form $f + g + h$, where f is smooth (with a known gradient), and g and h have a closed form proximal operator (even though $g + h$ does not), use the three operator splitting (TOS) method,

$$\begin{aligned} x &\leftarrow \text{prox}_{\gamma g}(z - \gamma u - \gamma \nabla f(x)) \\ z &\leftarrow \text{prox}_{\gamma h}(x + \gamma u) \\ u &\leftarrow u + \gamma^{-1}(x - z) \end{aligned}$$

where the step size γ is sufficiently small:

$$f(x) \leq f(z) + \langle \nabla f(z), x - z \rangle + \frac{1}{2\gamma} \|x - z\|^2$$

(i.e., such that the rhs be a quadratic majorizer of f).

Shampoo:
preconditioned stochastic tensor optimization
V. Gupta et al. (2018)

Preconditioning ignores the matrix or tensor structure of the parameters; Shampoo is a generalization of AdaGrad (a diagonal preconditioner) using a separate (dense) preconditioning matrix for each dimension of the gradient tensor. For matrices:

$$\begin{aligned} W_1 &= 0 & L_0 &= \varepsilon I & R_0 &= \varepsilon I \\ G_t &= \nabla f(W_t) & & & & \text{gradient} \\ L_t &\leftarrow L_{t-1} + G_t G_t' & & & & \text{preconditioners} \\ R_t &\leftarrow R_{t-1} + G_t' G_t & & & & \\ W_{t+1} &\leftarrow W_t - \eta L_t^{-1/4} G_t R^{-1/4} & & & & \text{parameters} \end{aligned}$$

Fairness without demographics
in repeated loss minimization
T.B. Hashimoto et al. (2018)

Minority groups contribute less to the loss function and suffer higher loss and higher attrition – disparity increases. *robust optimization* (minimizing the worst case risk over distributions close to the empirical ones for the χ^2 divergence,

$$D_{\chi^2}(P\|Q) = \int \left(\frac{dP}{dQ} - 1 \right)^2 dQ,$$

using the dual formulation) prevents *disparity amplification*.

Augmented CycleGAN: learning
many-to-many mappings from unpaired data
A. Almahairi et al. (2018)

CycleGAN learns 1-to-1 mappings. For 1-to-many mappings, add a latent space



and consider two generators and two encoders

$$\begin{aligned} A \times Z_b &\longrightarrow B \\ B \times Z_a &\longrightarrow A \\ A \times B &\longrightarrow Z_a \\ A \times B &\longrightarrow Z_b \end{aligned}$$

Is generator conditioning
causally related to GAN performance?
A. Odena et al. (2018)

To evaluate the quality of a GAN, check the *inception score*

$$\exp \mathbb{E}_{\text{image} \sim \text{generator}} \text{KL}(p(\text{label}|\text{image}) \| p(\text{label}))$$

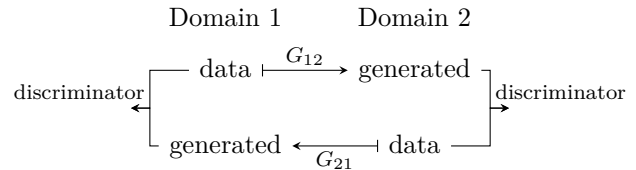
or the *Fréchet inception distance*, i.e., the Wasserstein distance W_2 between Gaussian fits of the activations,

$$W_2 = \|\mu_1 - \mu_2\|_2^2 + \text{tr}(V_1 + V_2 - 2(V_1 V_2)^{1/2}).$$

Poor quality is linked to the condition number of the Jacobian of the generator. *Jacobian clamping* feeds two minibatches to the generator, differing by small perturbations, computes the relative changes $\|\Delta G(z)\| / \|\Delta z\|$, and adds a penalty if they are not all in some predefined interval $[\lambda_{\min}, \lambda_{\max}]$.

MAGAN: aligning biological manifolds
M. Amodio (2018)

To align two domains, use two GANs



and add reconstruction losses for $G_{12} \circ G_{21} \approx \text{Id}$ and $G_{21} \circ G_{12} \approx \text{Id}$.

Learning longer-term dependencies in RNNs
with auxiliary losses
T.H. Trinh et al. (2018)

To have RNNs learn longer-term dependencies, use truncated BPTT with auxiliary losses, to forecast past (and/or future) observations.

Attention-based deep multiple instance learning
M. Ilse et al. (2018)

In *multiple instance learning* (MIL), labels are not assigned to instances but to bags of instances (labels may exist for instances, though, aggregated in each bag with a max or a sum). A function $S : \mathcal{P}(\text{Data}) \rightarrow \mathbf{R}$ is *symmetric* if it can be written $S(X) = g(\sum_{x \in X} f(x))$, for some functions f and g ; it can then be approximated arbitrarily close as $S(x) \approx g(\text{Max}_{x \in X} f(x))$. For a binary output, model f and g as neural nets and maximize the Bernoulli likelihood.

Mixed batches and symmetric discriminators
for GAN training
T. Lucas et al. (2018)

To reduce *mode collapse* in GANs, feed the discriminator a batch of true and generated samples and have it forecast the proportion of true samples.

Equivariant and invariant layers can be obtained as

$$\begin{aligned}(x_1, \dots, x_n) &\longmapsto (f x_1, \dots, f x_n) \\ (x_1, \dots, x_n) &\longmapsto A \rho(f x_1, \dots, f x_n)\end{aligned}$$

where ρ is the (elementwise) mean, standard deviation or maximum (or a basis of symmetric polynomials). The following layers suffice:

$$(x_1, \dots, x_n) \mapsto (\sigma(\beta + A x_1 + B \bar{x}), \dots, \sigma(\beta + A x_n + B \bar{x})).$$

CoVeR: learning covariate-specific vector representations with tensor decompositions

K. Tian et al. (2018)

The GloVe vector embedding can be generalized to deal with covariates

A : co-occurrence matrix

i, j : words

k : covariates (context)

$$f : \text{weight function, e.g., } 1 \text{ or } \left(\frac{\text{Min}(100, x)}{100} \right)^{.75}$$

$$\text{Argmin}_{v, c, b} \sum_{i, j, k} ((c_k \odot v_i)'(c_k \odot v_j) + b_{ik} + b_{jk} - \log A_{ijk})^2$$

Learning in integer latent variable models with nested automatic differentiation

D. Sheldon et al. (2018)

The forward algorithm, for hidden Markov models,

k : time

n_k : population size

z_{ik} : offspring of individual $i \in \llbracket 1, n_{k-1} \rrbracket$

m_k : immigration

$$n_k = \sum_{i=1}^{n_{k-1}} z_{k,i} + m_k \text{ (hidden)}$$

$$y_k \sim \text{Binomial}(n_k, p) \text{ (observed)}$$

can be formulated with probability generating functions.

Born-again neural networks

T. Furlanello et al. (2018)

Knowledge distillation trains a small (student) network to mimic a larger (teacher) network; try with two similarly-sized networks.

Differentiable dynamic programming for structured prediction and attention

A. Mensch and M. Blondel (2018)

To make dynamic programming (Viterbi, dynamic time warp (DTW)) differentiable, replace the (Max, +) semiring with (Max $_{\Omega}$, +)

$$\text{Max}_{\Omega}(x) = \text{Max}_{q \in \Delta^D} \langle q, x \rangle - \Omega(q)$$

$$\nabla \text{Max}_{\Omega}(x) = \text{Argmax}_{q \in \Delta^D} \langle q, x \rangle - \Omega(q)$$

where Ω is the negentropy $-H$ (Max $_{\Omega} = \log \text{sum exp}$, $\nabla \text{Max}_{\Omega} = \text{softmax}$), or the squared ℓ^2 norm ($\nabla \text{Max}_{\Omega}$ is then sparse).

Analysis of minimax error rate for crowdsourcing and its application to worker clustering model

H. Imamura et al. (2018)

The **Dawid and Skene** model of crowdsourcing uses the EM algorithm to estimate the confusion matrix of the workers $P[\text{worker } i \text{ gives label } k \text{ instead of } \ell]$ and the true labels.

Neural networks should be wide enough to learn disconnected decision regions

Q. Nguyen et al. (2018)

If no layer has more units than the input, the decision regions are connected (for ReLU activations).

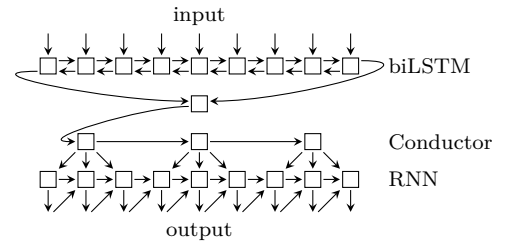
A hierarchical latent vector model for learning long-term structure in music

A. Roberts et al. (2018)

The second term in the ELBO

$$E[\log p(x|z)] - \text{KL}(q(z|x) \| p(z)) \leq p(x)$$

can be seen as a penalty and replaced with $\lambda \cdot \text{KL}$ or $(\text{KL} - \tau)_+$.



Learning long-term dependencies via Fourier recurrent units

J. Zhang et al.

Statistical recurrent units (SRU) keep moving averages of past hidden states; Fourier recurrent units keep their (truncated) Fourier transform.

Learning in reproducing kernel Kreĭn spaces

D. Oglic and T. Gärtner (2018)

A **Kreĭn space** is a vector space H with a non-degenerate symmetric but indefinite bilinear form admitting a decomposition $H = H_+ \oplus H_-$ into orthogonal Hilbert spaces such that $\langle f, g \rangle_H = \langle f_+, g_+ \rangle_{H_+} - \langle f_-, g_- \rangle_{H_-}$. The inner product $\langle \cdot, \cdot \rangle_{H_+} + \langle \cdot, \cdot \rangle_{H_-}$ makes H a Hilbert space; its topology does not depend on the choice of decomposition.

A *reproducing kernel Kreĭn space* is a Kreĭn space $K \subset \mathbf{R}^X$ for which the evaluation is continuous. If $k : X \times X \rightarrow \mathbf{R}$ is a difference of positive kernels (not every non-degenerate symmetric kernel is), it defines a reproducible kernel Kreĭn space.

Goodness-of-fit testing for discrete distributions via Stein discrepancy
J. Yang et al. (2018)

To characterize convergence in distribution towards p , the **Stein method** uses a “Stein operator” \mathcal{A}_p , usually defined from the score function of p (which can be computed from an unnormalized probability distribution), e.g.,

$$\mathcal{A}_p f(x) = \nabla \log p(x) f(x) + \nabla f(x),$$

and a set of test functions \mathcal{F} , e.g., $W^{2,\infty}$, or the unit ball of a RKHS, such that

$$\forall f \in \mathcal{F} \quad \mathbb{E}_{x \sim q} [\mathcal{A}_p f(x)] = 0 \quad \text{iff} \quad q = p.$$

The *Stein discrepancy*

$$\sup_{f \in \mathcal{F}} \mathbb{E}_{x \sim q} [\mathcal{A}_p f(x)]$$

can be used as a goodness of fit statistic for unnormalized distributions.

For discrete distributions, use

$$\Delta_i f(\mathbf{x}) = f(\mathbf{x}) - f(\sigma_i \mathbf{x})$$

$$\Delta_i^* f(\mathbf{x}) = f(\mathbf{x}) - f(\sigma_i^{-1} \mathbf{x})$$

$$s_i(\mathbf{x}) = \frac{\Delta_i p(\mathbf{x})}{p(\mathbf{x})}$$

$$\mathcal{A}_p f(\mathbf{x}) = s(\mathbf{x}) f(\mathbf{x})' - \Delta^* f(\mathbf{x})$$

where $\mathbf{x} \in X^d$, X is finite, σ is a cyclic permutation of X , σ_i only acts on the i th component, $f : X^d \rightarrow \mathbf{R}^m$, p is a distribution on X^d . For the unit ball of a RKHS,

$$D(q \| p) = \sup_f \mathbb{E}_{x \sim q} [\text{tr} \mathcal{A}_p f(x)]$$

can be computed as

$$D(q \| p) = \mathbb{E}_{x, y \sim q} [\kappa_p(x, y)].$$

A spline theory of deep networks
R. Balestrierio and R.G. Baraniuk (2018)

To make the features learned by a neural net more orthogonal, add a penalty

$$\lambda \cdot \sum_{i \neq j} |\langle w_i, w_j \rangle|^2.$$

Non-overlap-promoting variable selection
P. Xie et al. (2018)

The multilogistic model has a vector of parameters w_i for each class i . Easy-to-interpret sparse vectors with little overlap can be obtained with an ℓ^1 penalty and a penalty encouraging the vectors w_i to be orthogonal or orthonormal, *i.e.*, encouraging the Gram matrix $G_{ij} = w_i' w_j$ to be close to the identity.

The **log-determinant divergence** measures how close two matrices are.

$$D(X, Y) = \text{tr}(XY^{-1}) - \log \det(XY^{-1})$$

Variable selection via penalized neural network: a drop-out-one loss approach
M. Ye and Y. Sun (2018)

Variant of backward elimination (stepwise regression), with a neural network, for variable selection: fit a complete model, drop one variable *without re-fitting*, discard it if the change in loss is small.

Nonparametric variable importance using an augmented neural network with multitask learning
J. Feng et al. (2018)

To estimate (local) variable importance

$$(E[Y|X] - E[Y|X_{-i}])^2$$

in a neural network, augment its input with a mask indicating which variables to use.

Accelerated spectral ranking
A. Agarwal et al. (2018)

Pairwise or multiway comparisons (sports ranking, recommendation systems, social choice) can be aggregated: the Bradley-Terry model weights

$$P[i \succ j] = \frac{w_i}{w_i + w_j}$$

are the centralities of a graph, and can be estimated with random walks.

Ranking distributions based on noisy sorting
A. El Mesaoudi-Paul et al. (2018)

Define a probability distribution of \mathfrak{S}_n by using a sorting algorithm (insertion, quicksort) with comparisons replaced by Bernoulli trials; the Bernoulli parameters are pairwise preferences. Other distributions on \mathfrak{S}_n include:

- **Mallows:** $P[\sigma] \propto \exp -k\tau(\sigma)$, where τ is Kendall's tau (it can be centered on a permutation $\sigma_0 \neq \text{Id}$ by using $\tau(\sigma_0 \sigma \sigma_0^{-1})$);
- **Generalized Mallows:** $P[\sigma] \propto \exp - \sum k_i d_i(\sigma)$, where $d_i(\sigma) = \sum_{j>i} \mathbf{1}_{\sigma^{-1}(j) < \sigma^{-1}(i)}$ (it can be further generalized to account for a hierarchical structure on $\llbracket 1, n \rrbracket$);
- **Plackett-Luce:** $P[\sigma(i) = k | \sigma(1), \dots, \sigma(i-1)] \propto \theta_k$, *i.e.*,

$$P[\sigma] = \prod_i \frac{\theta_{\sigma^{-1}(i)}}{\theta_{\sigma^{-1}(i)} + \theta_{\sigma^{-1}(i+1)} + \dots + \theta_{\sigma^{-1}(n)}}$$

- **Babington-Smith:** the orderings of pairs are independent Bernoulli trials, rejected if they do not give an order relation,

$$P[\sigma] \propto \prod_{i < j} P_{\sigma^{-1}(i), \sigma^{-1}(j)},$$

A probabilistic theory of supervised similarity learning for pointwise ROC curve optimization
R. Vogel et al. (2018)

Pointwise ROC optimization maximizes the TPR (true positive rate) for a fixed FPR (false positive rate).

Pairwise bipartite ranking learns to recognize if two observations x, x' have the same label $y = y'$ (e.g., if two photographs are of the same person), $(x, x') \mapsto \mathbf{1}_{y=y'}$.

Fast stochastic AUC maximization with $O(1/n)$ convergence rate
M. Liu et al. (2018)

The AUC is often maximized using a surrogate square loss

$$\text{AUC} = P[f(x_1) < f(x_2) \mid y_1 = +1, y_2 = -1]$$

$$\text{Loss}(f) = E[(f(x_1) - f(x_2) - 1)^2 \mid y_1 = +1, y_2 = -1];$$

this can be turned into a saddle point problem.

Accelerated stochastic mirror descent: from continuous-time dynamics to discrete-time algorithms
P. Xu et al. (2018)

Stochastic mirror descent (SMD)

$$y_{k+1} \leftarrow \nabla h(x_k) - \eta \nabla f(x_k)$$

$$x_{k+1} \leftarrow \nabla h^*(y_{k+1})$$

minimizes $E[f]$, where h is convex on X , e.g., $h(x) = \|x\|_2^2$ on \mathbf{R}^2 or $h(x) = \sum x_i \log x_i$ on Δ_n , and $h^*(y) = \sup_{x \in X} \langle y, x \rangle - h(x)$.

Batch Bayesian optimization via multi-objective acquisition ensemble for automated analog circuit design
W. Lyu et al. (2018)

Multiobjective optimization of multiple acquisition functions leads to (single-objective) *batch* optimization.

Subspace embedding and linear regression with Orlicz norm
A. Andoni et al. (2018)

Orlicz norms generalize ℓ^p norms and (when used for losses rather than penalties) provide robust estimators (not unlike ℓ^p , $1 \leq p < 2$).

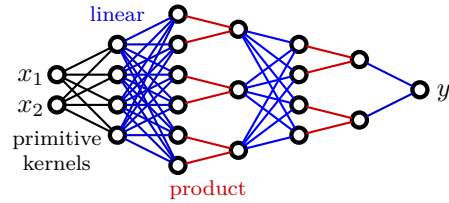
$$f(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq \delta \\ \delta(|x| - \frac{1}{2}\delta) & \text{otherwise} \end{cases} \quad (\text{Huber})$$

$$G(x) = f(f^{-1}(1)x)$$

$$\|x\|_G = \inf \left\{ \alpha > 0 : \sum G(|x_i|/\alpha) \leq 1 \right\}$$

Differentiable compositional kernel learning for Gaussian processes
S. Sun et al. (2018)

Since linear combinations and products of Gaussian process kernels are still kernels, a neural network, starting with primitive kernels and alternating those two operations, can be used to model them – this replaces the automated statistician’s discrete search on a context-free grammar with an end-to-end differentiable optimization problem.



Network global testing by counting graphlets
J. Jin et al. (2018)

The degree-corrected mixed membership model generalizes the *stochastic block model* (SBM)

$$P[A_{ij} = 1] = \theta_i \theta_j \sum_{1 \leq k, \ell \leq K} \pi_{ik} \pi_{j\ell} P_{k\ell},$$

where θ is the degree heterogeneity and π measures the mixed membership of the K communities; it can be written as $A = \Omega - \text{diag } \Omega + \text{noise}$, with $\Omega = \Theta \Pi \Pi' \Theta$.

The *graphlet count* statistic tests for $H_0 : K = 1$.

L_m : density of length- m self-avoiding paths

C_m : density of length- m self-avoiding cycles

$$\chi_m = C_m = (L_{m-1}/L_{m-2})^m \quad (m \geq 3)$$

Online convolutional sparse coding with sample-dependent dictionary
Y. Wang et al. (2018)

Sparse coding is a sparse matrix decomposition.

Given $X \in \mathbf{R}^{n \times N}$ (columns = observations)
Find $Y \in \mathbf{R}^{n \times d}$ (columns = filters)
 $W \in \mathbf{R}^{d \times N}$ (columns = weights)

$$\text{To minimize } \|X - YW\|_2^2 + \lambda \sum_i \|W_{\cdot i}\|_1$$

GAIN: missing data imputation using generative adversarial nets
J. Yoon et al. (2018)

GANs can be used to impute missing data: the generator fills in missing values, while the discriminator tries to find which values were imputed.

**Inter and intra topic structure learning
with word embeddings
H. Zhao et al. (2018)**

Hierarchical topic models generalizing *Poisson factor analysis* model texts as mixtures of subtopics, subtopics as mixtures of topics, and topics as distributions on words.

$$\begin{aligned} X &\sim \text{Poisson}(\Phi_1 \Theta_1) && (\text{words}) \\ \Theta_1 &\sim \text{Gamma}(\Phi_2 \Theta_2) \\ \Theta_2 &\sim \text{Gamma}(\Phi_3 \Theta_3) \\ &\vdots \\ \Phi_i &\sim \text{Dir}(\beta_i) \end{aligned}$$

They can also use a vector embedding f :

$$\beta \sim \text{Gamma}(\alpha, e^{f'w}).$$

**Improving regression performance
with distributional losses
E. Imani and M. White (2018)**

Do not forecast values $E[y|x]$ but distributions $y|x$, e.g., forecast the histogram density q_x to minimize $D(p_y||q_x)$, where $p_y = N(y, \sigma^2)$, with σ fixed.

**Comparison-based random forests
S. Haghir et al. (2018)**

A **comparison tree** is a decision tree whose boundaries are obtained by choosing two points x_1, x_2 with different labels and splitting the data with $d(x, x_1) < d(x, x_2)$.

To use a comparison random forest, one only needs to be able to actively tell if the new item A is closer to items B or C .

**QuantTree: histograms for change detection
in multivariate data streams
G. Boracchi et al. (2018)**

To detect distributional changes in multivariate time series, use histogram statistics, but build those histograms on kd-trees rather than grids.

**Deep learning
I. Goodfellow et al. (2016)**

3. The probability theory needed to deal with probabilistic graphical models boils down to two rules.

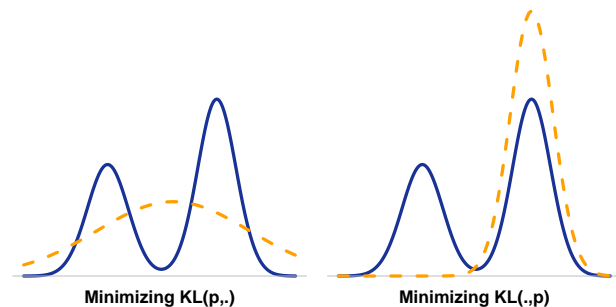
$$\begin{aligned} \text{Sum rule} & \quad P(x) = \sum_y P(x, y) \\ \text{Product rule} & \quad P(x, y) = P(x)P(y|x) \end{aligned}$$

The Kullback-Leibler divergence is the extra amount of information needed to send a message from P using

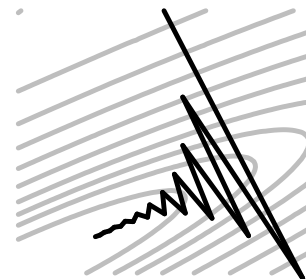
a code for Q .

$$\begin{aligned} \text{Information} & \quad I(x) = -\log p(x) \\ \text{Entropy} & \quad H(X) = E[-\log p(X)] \\ \text{KL divergence} & \quad D(P||Q) = E_{x \sim P} \log \frac{P(x)}{Q(x)} \geq 0 \\ \text{Cross-entropy} & \quad H(P, Q) = E_{x \sim P} [-\log Q(x)] \\ & \quad = H(P) + D(P||Q) \end{aligned}$$

Log-likelihood and cross-entropy are the same thing, and the Kullback-Leibler divergence only differs by a constant: minimizing $D(\text{data}||\text{model})$ is equivalent to minimizing $H(\text{data}, \text{model})$.



4. Gradient descent fails to exploit the curvature information in the Hessian – it “wastes time repeatedly descending canyon walls because they are the steepest feature”.



5. Balancing over- and under-fitting requires to both make the training error small and the gap between training and test error small.

The **Bayes error** is the error made by the best predictor possible, *i.e.*, that knowing the true data generation process (the error is not zero because of the error in the data generation process): it is the minimum test set error.

To fight the no free lunch theorem, incorporate preferences into the learning algorithm (prior, regularization).

There are implicit priors in many machine learning algorithms: locally constant (k -means), locally smooth, but deeper algorithms allow for more complex priors – non-local generalization, hierarchical structures.

6. You may need to clip gradients for *mixture density networks* (MLP with Gaussian mixtures in their output).

From the *universal approximation theorem*, one layer is enough, but deeper networks require fewer units.

7. Regularization trades bias for variance. Neural networks may use a separate penalty for each layer.

L^2 regularization shrinks towards zero, but more in the “imprecise” directions (where the Hessian eigenvalues are large). Early stopping, for a linear model, is equivalent to L^2 regularization.



L^1 regularization has a sparsifying effect.

You can add noise to:

- Input (denoising auto-encoder);
- Activations (dropout corresponds to multiplicative noise);
- Weights (this makes the network Bayesian);
- Output (label smoothing replaces 0 and 1 with $\varepsilon/(k-1)$ and $1-\varepsilon$).

Parameter sharing forces parameters to be equal (CNN) or close (transfer learning).

To predict with dropout, you could average over several masks; alternatively, just rescale the weights (there is no theoretical justification for nonlinear models: it is an approximation, but it works well). Multiplying the weights by a random number $\mu \sim N(1,1)$ is another form of dropout, but since $E[\mu] = 1$, no rescaling is necessary.

If you do not have enough data, prefer Bayesian neural nets or semisupervised learning.

To generalize well, neural nets should be locally constant around (orthogonally to) the data manifold – instead, they tend to be linear. Try data augmentation with adversarial examples, or a penalty for $\|\nabla_{\theta} f\|$.

8. The ideal objective function, the expected loss over the data generation process, is not available: instead, we minimize the *empirical risk*, *i.e.*, the average loss over the training set. Non-differentiable loss functions (e.g., 0-1 loss) are often replaced by differentiable surrogates (e.g., negative log-likelihood). Optimization algorithms often converge faster with *mini-batches* ($n \geq 100$ for gradient-based methods, $n \geq 10,000$ for Hessian-based ones), *i.e.*, with frequent though imprecise updates. The mini-batches should be selected randomly.

Gradient descent may not arrive at a critical point – plot the gradient norm: it often fails to decrease. The concern that we may end up at a local minimum or a saddle point is moot.

Models with latent variables are not identifiable: they have many (sometimes uncountably many) local minima – but most minima have a low cost value.

Cliffs in the loss function landscape can be avoided with *gradient clipping*.

Momentum addresses the high variance of the stochastic gradient and bad conditioning of the Hessian. *Nesterov momentum* computes the gradient after the momentum step.

Weight initialization is usually random (to break symmetries) and normalized.

AdaGrad and *RMSProp* decrease the learning rate using the sum or exponential moving average of past gradient norms; *Adam* combines momentum and RMS-Prop.

Second-order methods often assume the Hessian is positive definite, which is not the case around saddle points: the Hessian can be regularized, $H \rightsquigarrow H + \lambda I$ (Levenberg-Marquardt). With gradient descent (and line search), each gradient direction is perpendicular to the previous one, resulting in a zigzag motion. *Conjugate gradient* (CG) replaces this orthogonality condition $d_t' d_{t-1} = 0$ with $d_t' H d_{t-1} = 0$, where $d_t = \nabla J + \beta_t d_{t-1}$ ($\beta_t = 0$ gives the gradient). Some approximations of β_t do not require the Hessian, e.g.,

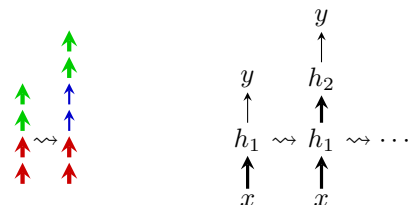
$$\beta = \frac{\text{current } (\nabla J)'(\nabla J)}{\text{previous } (\nabla J)'(\nabla J)}$$

$$\text{or } \beta = \frac{(\text{change in } \nabla J)'(\text{current } \nabla J)}{\text{previous } (\nabla J)'(\nabla J)}.$$

The gradients tell how to update each parameter if the others remain fixed; *batch normalization* helps decouple the layers.

Polyak averaging (Cesàro summation) can help convergence.

Greedy supervised pretraining builds up an increasingly deeper network.

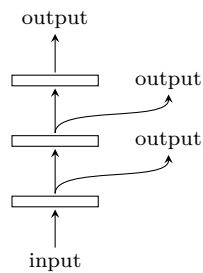


FitNets train a wide and shallow “teacher” network, and then a deeper “student” network, to forecast both the output and the middle layer of the teacher network, providing hints for its middle layers.

Gradients flow more easily through linear functions: ReLU are easier to train than sigmoids.

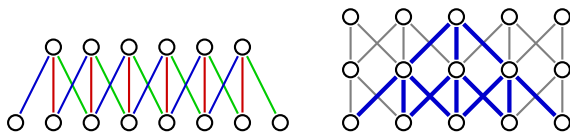
Skip-connections (ResNets) reduce the length of the shortest path to the output.

Auxilliary heads are an alternative to pre-training.



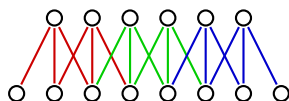
Continuation methods use a sequence of increasingly more difficult loss functions (blurring non-convex functions makes them more convex); *curriculum learning* (e.g., stochastic curriculum learning, with a mixture of easy and hard samples), is an example.

9. Convolution (same notion as in math, but without flipping the kernel) can be expressed as multiplication by a structured matrix (circulant and sparse). CNNs are sparsely connected: each neuron is only linked to nearby neurons (small receptive field – but it grows with depth).

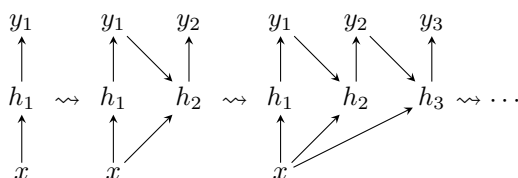


Max pooling can be interpreted as a summary statistic of nearby inputs, or as a prior that the output of the layer should be invariant under small translations.

Locally-connected layers are sparsely connected, but do not share weights. *Tiled convolutions* only share weights locally.



“Recurrent” convolutional networks are progressively refined.



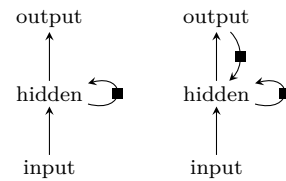
For separable kernels, the convolution can be computed separately in each dimension.

To speed up training, skip it, and use random features, hand-crafted features, features from k -means clustering of image patches, greedy layer-wise pretraining.

Real-world convnets use more branching.

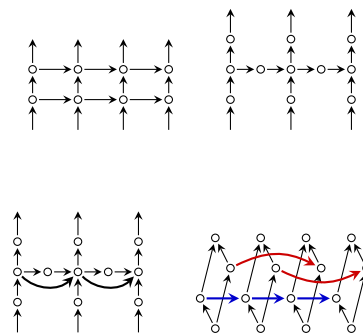
10. One-dimensional convolution provides parameter sharing, but it is shallow: each output only depends on the recent past. Recurrent neural nets (RNN) use

a different form of sharing. Back-propagation through time (BPTT) is backpropagation on the unrolled network.



Instead of feeding \hat{y}_t to the network at time $t + 1$, *teacher forcing* uses y_t (during training – otherwise it is not available) and progressively switches to actual forecasts.

Deep recurrent networks can stack layers vertically or horizontally, use skip connections, or remove connections.



Echo state networks (reservoir computing) are similar to kernel machines.

The linear maps $x \mapsto Wx$ can be replaced with bilinear (product) or multilinear (tensor) ones.

Recursive neural nets are tree-shaped neural nets.

Attention and bidirectional RNNs can help.

To deal with exploding gradients, clip them (or take a random step when they are too large); for vanishing gradients, use LSTMs. Try adding a penalty to keep the gradient norm approximately constant.

To improve memory, add skip connections, remove some short connections, and use leaky memory, $m_t \leftarrow (1 - \lambda)m_{t-1} + \lambda v_t$, with different time scales λ .

If LSTMs and attention do not provide enough memory, add explicit memory (neural Turing machine, NTM).

11. Learning rate is the most important parameter.

Increase capacity to reduce training error; increase regularization to reduce the gap between training and testing errors.

Visualize the worst mistakes; fit a tiny dataset; monitor histograms of activations and gradients (parameter updates should be around 1% of the parameter values).

Allow the model to refuse to make a decision if it is too uncertain.

12. Cascades (using a fast high-recall model followed by a more expensive high-precision one) are a form of hard attention.

In an *expert network*, a top-level model decides which expert (model) or, better, combination of experts to use.

Beware of low-contrast, uninformative images when using contrast normalization.

NLP inputs can be preprocessed with n -grams, smoothing, or word embeddings. The hierarchical softmax can help generate high-dimensional outputs.

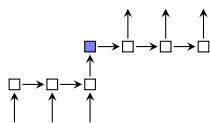
The gradient of a language model can be decomposed as

$$\begin{aligned}\frac{\partial}{\partial \theta} \log \frac{\exp a_y}{\sum_i \exp a_i} &= \frac{\partial a_y}{\partial \theta} - \sum p_i \frac{\partial a_i}{\partial \theta} \\ &= \frac{\partial a_y}{\partial \theta} - E_i \left[\frac{\partial a_i}{\partial \theta} \right]\end{aligned}$$

(positive and negative phases) and the second term, a huge sum, can be approximated by sampling (but this requires knowing all the $p_i = P[Y = i]$), importance sampling (but this still requires the p_i/q_i) or *biased importance sampling* (normalize the importance weights to sum to 1).

If the output is a bag of words, minimize the loss for the positive words and an equal number of negative words (sampled randomly, with a focus on words likely to be mistaken, and importance weights). Alternatively, forecast a score rather than a probability, and make that of the correct output at least one more than those of the incorrect ones.

Recurrent neural nets (RNNs) for translation first learn a sentence embedding, and then generate a translation, often with an attention mechanism.



Use t-SNE to plot word embeddings.

Collaborative filtering is a low-rank matrix factorization (SVD with missing values), *Content-based recommendation systems* address the cold-start problem by learning an embedding of user and/or item features. Recommender systems are contextual bandits.

Other applications include embeddings (of words, phrases, relations, etc.), and learning structured data (knowledge base) for link prediction and question answering.

13. Nor all learning problems are supervised: generative models, outlier/anomaly detection, missing values, semi-supervised learning, unsupervised learning.

PCA has many variants.

PCA	$X \sim N(b, WW')$
Probabilistic PCA	$X \sim N(b, WW' + \sigma^2 I)$
Factor analysis	$X \sim N(b, WW' + \Delta)$

Equivalently,

$$\begin{aligned}h &\sim N(0, I) & h &\sim N(0, I) & h &\sim N(0, I) \\ X = b + Wh & & \varepsilon &\sim N(0, \sigma^2 I) & \varepsilon &\sim N(0, \Delta) \\ & & X &= b + Wh & X &= b + Wh\end{aligned}$$

$$\begin{aligned}h &\sim p \text{ non-Gaussian} \\ X &= b + Wh \quad W \text{ orthogonal (ICA)}\end{aligned}$$

Nonlinear variants replace $h \mapsto b + Wh$ with $h \mapsto f(h)$, a nonlinear mapping given by, e.g., a neural net.

Slow feature analysis generalizes PCA

$$\begin{aligned}\text{Find} & \quad u \in \mathbf{R}^n \\ \text{To maximize} & \quad E(u'X)^2 \\ \text{Such that} & \quad u'u = 1\end{aligned}$$

to time series

$$\begin{aligned}\text{Find} & \quad u \in \mathbf{R}^n \\ \text{To maximize} & \quad E(u'x_{t+1} - u'x_t)^2 \\ \text{Such that} & \quad E[u'x_t] = 0 \\ & \quad E(u'x_t)^2 = 1.\end{aligned}$$

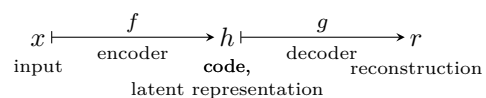
The mapping $x \mapsto u'x$ can be replaced with an affine transform $x \mapsto u'x + b$. To learn several features, add a constraint $E[(u'_i x_t)(u'_j x_t)] = \delta_{ij}$. Apply on a nonlinear transform (e.g., a quadratic kernel) of x .

Sparse coding generalizes PPCA by using a prior with a sharp peak at 0 (not sparse, but still sparsifying).

$$\begin{aligned}h &\sim \text{Laplace, Cauchy or Student} \\ \varepsilon &\sim N(0, \sigma^2 I) \\ X &= b + Wh + \varepsilon\end{aligned}$$

Sparse coding can learn good features, but not how to combine them.

14. Autoencoders are nonlinear generalizations of PCA.



One can add a regularizer on the local features. To avoid trivial solutions ($f = \varepsilon \text{Id}$, $g = \varepsilon^{-1} \text{Id}$), impose $g = f^T$.

Contrastive autoencoders (CAE) add a penalty on $\|\nabla_{\text{input}} \text{latent}\|^2$ – large derivatives will only be kept if they are tangent to the data manifold.

In a *stochastic autoencoder* (SAE), the decoding function is replaced by a probability distribution

$$\text{input} \xrightarrow{f} \text{latent} \xrightarrow{p_\theta(\text{output}|\text{latent})} \text{output}$$

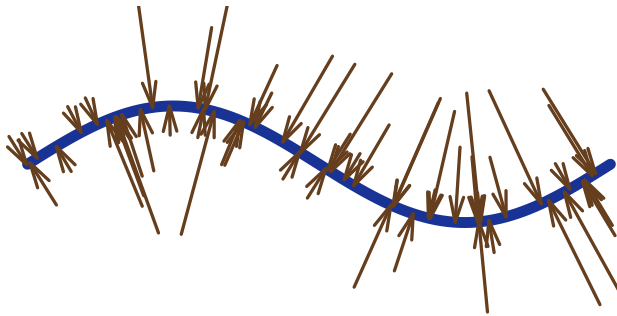
(f and p_θ are learnt). Both steps could be stochastic.

encoder: $p_{\theta_1}(h|x)$
decoder: $p_{\theta_2}(x|h)$

Denoising autoencoders (DAE) try to reconstruct their input from a noisy version of it:

$$\text{Maximize}_{p,f} \mathbb{E}_{x \sim \text{data}} \mathbb{E}_{\varepsilon \sim \text{noise}} p[x|h = f(x + \varepsilon)].$$

It is not possible to remove the noise if $x + \varepsilon$ is still on the data manifold, but if it is away from it, the denoising autoencoder will project it onto the data manifold: it learns a vector field that moves towards the manifold, $\nabla_x \log p_{\text{data}}(x)$.



Predictive sparse decomposition (PSD)

$$\begin{aligned} x &\mapsto f(x) \\ h &\mapsto g(h) \end{aligned}$$

minimizes (alternatively h and f, g in)

$$\|x - g(h)\|^2 + \lambda \|h\|_1 + \gamma \|h - f(x)\|^2.$$

15. *Unsupervised (greedy, layerwise) pretraining* is sometimes helpful (e.g., when there is not enough data), but often detrimental (it assumes some features useful for the unsupervised task are also useful for the supervised one; it relies on a regularizing effect of the choice of starting point to avoid “bad local minima” – but those are extremely rare and this could drastically reduce exploration). Prefer dropout and batchnorm (big data) or Bayesian methods (small data) – unsupervised pretraining is only still useful in NLP.

Transfer learning (same input, different tasks) and *domain adaptation* (different inputs, same tasks) are forms of *supervised pretraining*. One-shot learning uses a single labeled example for the new task. Zero-shot learning uses none, e.g.:


- Given textual descriptions of animals and photographs of some of them, learn to recognize those you have never seen;
- Given models for languages A and B and a bilingual $A \leftrightarrow B$ lexicon, learn to translate sentences.



Semi-supervised learning looks for a representation of the input x that disentangles its causes, in the hope that the output y is one of them – this will not work if there is no structure in x , e.g., $x \sim N(0, 1)$.

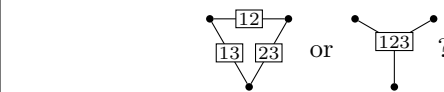
16. An *energy-based model* (or *Boltzman machine*) is a probability distribution of the form $p(x) \propto \exp -E(x)$ (a *Boltzman distribution* – it just says that $\forall x p(x) \neq 0$). In a *product of experts*, the energy is a sum of “experts”, each checking if some constraint is satisfied, using a subset of the variables.

Undirected models cannot represent “immoral” dependencies; directed models cannot represent “non-chordal” dependencies.

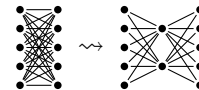


Undirected models are ambiguous: does  represent

 or , i.e., in terms of *factor graphs*

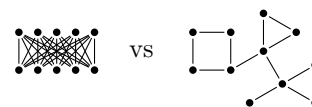


Directed models can be sampled by *ancestral sampling* (after choosing a topological ordering of the nodes). Undirected models can be sampled, approximately, by Gibbs sampling. Introducing hidden variables can replace many dependencies.



In general, computing the marginal probability of a probabilistic graphical model is #P-hard (NP problems are decision problems, #P problems are the corresponding counting problems); they can be approximated by variational inference.

Loopy belief propagation is not supposed to work with arbitrary graphs (non-trees), but works well with sparsely-connected graphs – neural networks are densely connected.



Restricted (i.e., bipartite) Boltzman machines (RBM) can be efficiently sampled from (block Gibbs sampling).

17. Expectations can be approximated with averages:

$$\mathbb{E}_{x \sim p} f(x) \approx \frac{1}{n} \sum_{x_i \sim p} f(x_i)$$

or weighted averages (*importance sampling*)

$$\begin{aligned} \mathbb{E}_{x \sim p} f(x) &= \int f(x)p(x)dx = \int \frac{f(x)p(x)}{q(x)}q(x)dx \\ &= \mathbb{E}_{x \sim q} \frac{f(x)p(x)}{q(x)} \\ &\approx \frac{1}{n} \sum_{x_i \sim q} \frac{f(x_i)p(x_i)}{q(x_i)} \end{aligned}$$

whose variance can be lower (it is minimal for $q \propto p \cdot |f|$) or *biased importance sampling*

$$\mathbb{E}_{x \sim p} f(x) \approx \sum_{x_i \sim q} \frac{\tilde{p}(x_i)}{\tilde{q}(x_i)} f(x_i) \Big/ \sum_{x_i \sim q} \frac{\tilde{p}(x_i)}{\tilde{q}(x_i)}$$

(with unnormalized probabilities, $p \propto \tilde{p}$, $q \propto \tilde{q}$ – the estimator is only asymptotically unbiased).

For MCMC, use 100 chains. Mixing time is determined by the second eigenvalue (the first is 1 – Perron-Frobenius).

Gibbs sampling has mixing problems in presence of several modes: increase the temperature (*tempering*). MCMC is less useful when the distribution has a manifold structure: good learning leads to high $\text{MI}(h, x)$, low entropy $p(x|h)$, which is hard to sample from and leads to bad mixing

18. To compute the gradient of the likelihood, we need that of the partition function (*negative phase* – the other tem, the unnormalized log-likelihood, is the positive phase).

$$p = \frac{\tilde{p}}{Z} \quad \nabla_{\theta} \log p = \nabla_{\theta} \log \tilde{p} - \nabla_{\theta} \log Z$$

It can be estimated with Monte Carlo:

$$\begin{aligned} \nabla_{\theta} \log Z &= \frac{\nabla_{\theta} Z}{Z} \\ &= \frac{\sum \nabla_{\theta} \tilde{p}(x)}{Z} \\ &= \frac{\sum \nabla_{\theta} \exp \log \tilde{p}(x)}{Z} \\ &= \sum p(x) \nabla_{\theta} \log \tilde{p}(x) \\ &= \mathbb{E}_{x \sim p} \nabla_{\theta} \log \tilde{p}(x). \end{aligned}$$

The positive phase increases \tilde{p} for x drawn from the data; the negative phase decreases $\tilde{p}(x)$ for x drawn from the model.

While $\nabla_{\theta} \log Z = \mathbb{E} \nabla_{\theta} \log \tilde{p}(x)$ could be estimated with MCMC, with a burn-in period, this would have to be done anew at each gradient step. *Contrastive divergence* (CD) uses k Gibbs steps, with no burn-in, after initialization from the data – but it is unaware of high-probability regions away from the data. It only works for shallow models (e.g., RBM) – the latent variables are not in the data and require a burn-in period. CD is not the gradient of any function (there could be cycles)

Stochastic maximum likelihood is CD, but it initializes the chain from the previous gradient; the gradient steps should be small and/or the chain long. *Parallel tempreing* can help.

Pseudo-likelihood uses

$$\log p(\mathbf{x}) = \sum \log p(x_i | x_1 \cdots x_{i-1}) \approx \sum \log p(x_i | \mathbf{x}_{-i})$$

(generalized pseudo-likelihood puts the variables in groups).

Score matching minimizes the expectation of

$$\begin{aligned} L(x, \theta) &= \frac{1}{2} \|\nabla_x \log p_{\text{model}}(x, \theta) - \nabla_x \log p_{\text{data}}(x)\|_2^2 \\ &= \sum_i \frac{\partial^2}{\partial x_i^2} \log p(x, \theta) + \frac{1}{2} \left(\frac{\partial}{\partial x_i} \log p(x, \theta) \right)^2 \end{aligned}$$

(Z does not depend on x and disappears; p_{data} is not known).

For binary data, *ratio matching* minimizes the average of

$$L(x, \theta) = \sum_i \left(1 + \frac{p(x, \theta)}{p(f_j(x, \theta))} \right)^2$$

where f_i flips the i th bit.

Noise contrastive estimation (NCE) $\log p(x) = \log \tilde{p}_{\theta}(x) + c$ considers the partition function $c = -\log Z$ as another parameter to estimate, not with maximum likelihood, but with a binary classifier, data vs noise (e.g., word2vec). For *self-contrastive estimation*, the noise samples are generated from the model – the model learns to distinguish reality from its own evolving beliefs.

To evaluate or compare models, with the likelihood or the BIC, the whole likelihood, with the partition function, is needed. The partition function can be estimated by *importance sampling*,

$$\begin{aligned} \frac{Z_1}{Z_0} &= \frac{1}{Z_0} \int \tilde{p}_1 = \frac{1}{Z_0} \int p_0 \frac{\tilde{p}_1}{p_0} = \int p_0 \frac{\tilde{p}_1}{\tilde{p}_0} \\ &= \mathbb{E}_{x \sim p_0} \frac{\tilde{p}_1(x)}{\tilde{p}_0(x)} \\ &\approx \frac{1}{N} \sum_{x_i \sim p_0} \frac{\tilde{p}_1(x_i)}{\tilde{p}_0(x_i)}. \end{aligned}$$

If the distributions are very different, *annealed importance sampling* (AIS) adds more distributions between p_0 (tractable) and p_1 (interesting), e.g., weighted geometric averages $p_0^{\eta_i} p_1^{1-\eta_i}$, and computes the successive ratios of partition functions.

19. The **evidence lower bound** (ELBO) is

$$\begin{aligned} \mathcal{L}(v, \theta, q) &= \mathbb{E}_{h \sim q} [\log p(h, v)] + H(q) \\ &= \log p(v) - D_{\text{KL}}(q(h|v) \| p(h|v)) \leq \log p \end{aligned}$$

where v are the observed variables, h the latent variables, q some distribution on h and $D_{\text{KL}} \geq 0$.

MAP estimation is equivalent to variational inference (VI) with q Dirac.

The *mean field approximation* uses a factorized q , $q(h|v) = \prod_i q(h_i|v)$ (it is easy for discrete variables but requires calculus of variations for continuous ones).

20. Learning in a Boltzman machine

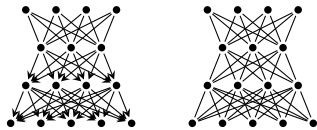
$$E(v, h) = - \begin{pmatrix} v \\ h \end{pmatrix}' \begin{pmatrix} R & \frac{1}{2}W \\ \frac{1}{2}W & S \end{pmatrix} \begin{pmatrix} v \\ h \end{pmatrix} - \begin{pmatrix} b \\ c \end{pmatrix}' \begin{pmatrix} v \\ h \end{pmatrix}$$

is local. Conditional distributions for RBM are factorial and easy to compute and sample from.

$$p(h|v) = \prod_j \sigma((2h - 1) \odot (c + W'v))_j$$

$$p(v|h) = \prod_i \sigma((2v - 1) \odot (b + Wh))_i$$

A *deep belief network* is a RBM with additional directed layers; a deep Boltzman machine only has undirected connections.



RBMs can be generalized to continuous variables: with *Gaussian-Bernoulli RBMs*, the mean (or both the mean and the variance) of the Gaussian output depends on the binary latent variables. The *mean-product of Student's T distribution* (mPoT) model replaces the Bernoulli latent variables with Gamma ones.

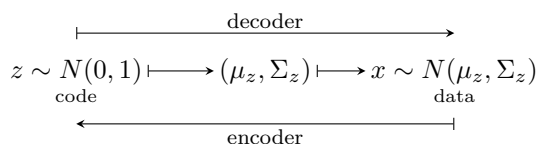
To compute the gradient in presence of random operations, e.g., ∂_μ or ∂_σ in $y \sim N(\mu, \sigma^2)$, rewrite the model as a transformation of samples from a fixed distribution, e.g.,

$$\varepsilon \sim N(0, 1)$$

$$y = \mu + \sigma \varepsilon$$

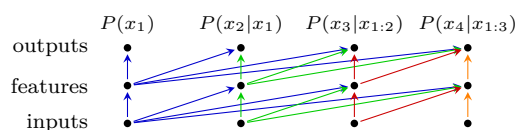
and consider the random Gaussian input as other inputs (“reparametrization trick”, “stochastic back-propagation”). For discrete distributions, it is trickier, but doable with reinforcement learning’s REINFORCE algorithm.

Variational autoencoders (VAE) combine a generator with an inference network.

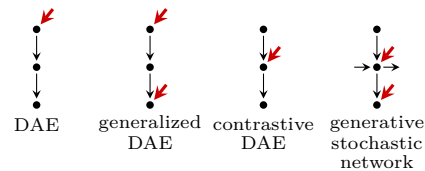


Generative adversarial networks (GAN) combine a generator with a discriminator.

Neural autoregressive networks reuse their features.



Variants of auto-encoders differ in where they add noise.



R in Finance 2018

Bitcoin. Random forests and gradient boosting can help forecast Bitcoin prices from price, volume, users, FX, VIX and search data.

Dynamic model averaging (DMA) combines several models (here, univariate cryptocurrency forecasts, from stock indices, commodities, interest rates, CDS, volatility), with time-varying weights. Implementation in **eDMA**.

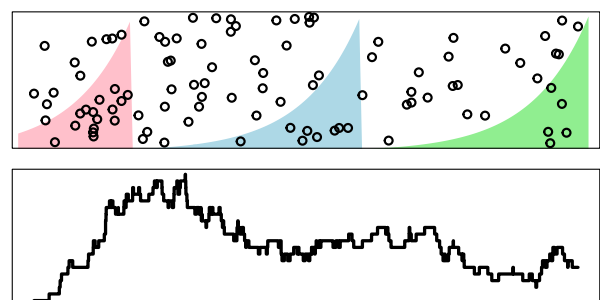
Bittrex is the 10th largest bitcoin exchange (4th in terms of number of trades or number of currencies), with a 2% market share it has a REST API and an R package. There are almost 300 cryptocurrencies (on this exchange), but their **cointegration dimension** is at most 4.

The **rbtc** package lets you run a Bitcoin node from R.

Time series models. The finite tick size makes prices discrete: they can be modelled with a **Skellam process** (a difference of Poisson processes) or, more generally, an integer-valued Lévy process. An integer-valued **trawl process** can be sampled as follows:

- Take points at random on $\mathbf{R} \times [0, 1]$ (Lévy basis – it could be a random signed measure, instead, if you are familiar with Lévy processes);
- Progressively move the “trawl” (a shape, e.g., a rectangle, or an exponential) and return the number of points in it.

The non-rectangular trawl makes changes more fleeting.



Avoid the **Hodrick-Prescott** (HP) filter:

- It creates spurious dynamic relations (autocorrelation, predictability), not only in the “trend” component (that is intended and desirable), but also in the “cycle” (noise) one;
- The results at the extremities of the interval are not usable – problematic if we want to use it on a moving or expanding window;

- It is often estimated with a Kalman filter

$$\begin{array}{ll} x_t = y_t + \varepsilon_t & \text{Observation} \\ y_t - 2y_{t-1} + y_{t-2} = \eta_t & \text{Evolution} \end{array}$$

but the estimated smoothing parameter is at odds with common practice.

Instead, just estimate a linear regression

$$y_{t+h} \sim y_t + y_{t-1} + y_{t-2} + y_{t-3}$$

(with $h = 8$ for quarterly data).

The time evolution of the value-at-risk (VaR) can be modeled directly, in an autoregressive way to account for volatility clustering, e.g.,

$$\text{VaR}_t = \beta_0 + \sum_i \beta_i \text{VaR}_{t-i} + \sum_j \gamma_j |x_{t-j}|$$

(or some other function of VaR_{t-i} , x_{t-j} and exogenous variables) and estimated by quantile regression.

The correlated idiosyncratic volatility shock model is yet another multivariate GARCH (factor) model.

A tidyverse for time series may be emerging: **tidyquant** (stock and portfolio analysis), **anomalize** (anomaly detection), **tibbletime** (**collapse_by**, **rollify**, **filter_time**, **as_period**), **timek** (time series machine learning), **sweep**, **furrr** (**future** + **purrr**), **flyingfox** (Quantopian's Zipline Python library via **reticulate**).

The *stationary bootstrap* is a block bootstrap with random block lengths.

Robust statistics. We may want to try to replace some of our location and dispersion estimators with those in the **RobStatTM** package:

- **RobStatTM::MLocDis(x)\$mv** (location and dispersion),
- **RobStatTM::lmrobdetMM**,
- **RobStatTM::lmrobdetDCML** (regression),
- **RobStatTM::MultiRobv(x, type="auto")** (covariance matrix).

Replacing the Fama-French cross-sectional regressions with robust regressions:

- Confirms the relation between B/P, earnings, size and future returns;
- Reveals a negative relation between beta and future returns;
- Suggests interactions (especially with size) are stronger than earlier thought.

Given a robust estimator θ_{rob} , with high breakdown point and good *asymptotic efficiency* $\text{Var}[\text{MLE}] / \text{Var}[\theta_{\text{rob}}]$ but poor finite-sample efficiency, the **distance-constrained maximum likelihood** (DCML) estimator shrinks it towards the maximum likelihood estimator.

$$\begin{array}{lll} \text{Find} & \theta & \\ \text{To minimize} & \text{KL}(\text{data} \| f_\theta) & \text{log-likelihood} \\ \text{Such that} & \text{KL}(f_{\theta_{\text{rob}}} \| f_\theta) \leq \delta & \end{array}$$

Shrinkage estimators of the variance matrix are not robust: instead of shrinking the sample variance matrix towards the identity, shrink the (robust) MCD variance matrix (the variance matrix estimated on the 50% or 75% of the data giving the lowest determinant), just enough for the matrix to be well-conditioned ($\lambda_{\max}/\lambda_{\min} < 1000$). Implementation in **robustbase::covMcd**, **rrcov**, **RiskPortfolios::covEstimation**.

Error estimation. The **influence function** of an estimator T on a distribution F is

$$\text{IF}(r, T, F) = \left. \frac{d[(1-\gamma)F + \gamma\delta_r]}{d\gamma} \right|_{\gamma=0}$$

The variance of the estimator can be estimated as

$$\text{Var}[T(F_n)] \approx \frac{1}{n} \sum \text{IF}^2(r_t, T, F_n)$$

The **Newey-West** standard error estimator gives the standard error of the sample mean in presence of autocorrelation

$$\hat{\sigma}_n[\hat{\mu}_n] = \frac{1}{\sqrt{n}} \left[C_0^2 + 2 \sum_{k \geq 1} \left(1 - \frac{k}{n}\right) C_k \right]$$

where C_k is the lag- k autocovariance (C_0 is the variance); the sum is often truncated to

$$\sum_{k=1}^m \left(1 - \frac{k}{m}\right) C_k,$$

for $m \ll n$. These results can be combined:

$$\text{Var } T(F_n) = E[\text{IF}^2(r_0)] + 2 \sum_{k \geq 1} E[\text{IF}(r_0)\text{IF}(r_k)].$$

Applying this formula to get an approximate finite-sample standard error of performance measures (value-at-risk, expected shortfall, Sharpe ratio) is problematic, but it turns out to be the spectral density at frequency zero,

$$\text{Var } T(F_n) = S_{\text{IF}}(0),$$

which can be estimated from the periodogram, as the intercept of a polynomial regression with exponential noise (the periodogram is exponentially-distributed), with a sparsity penalty (elastic net).

To test if a risk (VaR) model is correct, test if the *probability integral transform* (PIT)

$$P_t = F_t(\text{Loss}_t),$$

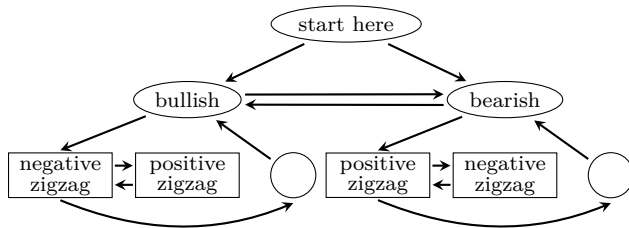
i.e., the running quantile, is $U(0, 1)$. Since accuracy in the “bad tail” (losses) is more important, test instead

$$H_0 : \nu(P_t) \sim \nu(U(0, 1)),$$

where ν is monotonic. Implementation in **spectralBacktest**.

The negative effects of errors in mean returns on the mean-variance efficient frontier increase with risk tolerance and number of assets.

Bayesian statistics. Hamiltonian Monte Carlo (HMC) cannot sample from discrete variables, but Stan is flexible and lets you implement the forward (filtering), forward-backward (smoothing) and Viterbi (MAP) algorithms yourself. A **hierarchical hidden Markov model** (HHMM) is a HMM with two types of nodes, emitting and non-emitting, arranged in layers.



An *input-output hidden Markov model* is an HMM whose transition and emission probabilities depend on an input (“control”) signal. Useful features, to forecast prices with a HHMM, include the presence of a local maximum or minimum, trends ($x_{n-4} < x_{n-2} < x_n \wedge x_{n-3} < x_{n-1}$), volume changes ($\log(v_n/v_{n-1})$, $\log(v_{n-1}/v_{n-2})$, $\log(v_n/v_{n-2})$, $\cdot > \alpha$, $\cdot < -\alpha$, $|\cdot| \leq \alpha$).

Stan can fit Gaussian processes.

Jags can estimate vine copulas.

Multilevel Monte Carlo (MLMC) combines Monte Carlo simulations with different timesteps $2^{-k}T$:

$$E[f_n(T)] = E[f_0(T)] + \sum_{k=1}^n E[f_k(T) - f_{k-1}(T)].$$

IT. The **partools** package provides MPI-like distributed computing with a higher-level interface.

The **foreach** package can not only parallelize your code via **multicore** or **parallel**, but also HPC schedulers (LSF, OpenLava, Slurm, with **foreach::registerdoFuture**) or Azure (**registerAzureParallel**).

Postgres lets you define functions (Γ), aggregate functions (Weibull distribution fit, Shapiro-Wilk test, **moments::skewness**, GARCH fit, etc.) and window functions (ARIMA forecast) in R.

The **hdf5r** package replaces the **h5** package presented two years ago [which did not work well].

Besides the obligatory Rcpp tutorial, there was also a Tensorflow one. The ecosystem around Tensorflow is growing. For instance, it is possible to write statistical models in R (rather than Stan/Jags/Bugs) and estimate them by MCMC via TensorFlow.

It is possible to launch CUDA kernels from Rcpp.

Options and trading. Here are a few volatility investing strategies:

- Long vol (VXX) when $\text{SMA}(\text{VIX}/\text{VXV}, 10) > 1$, short vol (XIV/SVXY) otherwise
- Switch between the highest 83-day momentum between XIV/SVXY, ZIV, VXZ, and VXX

- Long XIV/SVXY when $\text{SMA}(\text{VIX} - 10\text{day SPY vol}, 5) > 0$, long VXX otherwise
- Long XIV/SVXY when $\text{VIX}/\text{VXV} < .917$; long VXX when $\text{VIX}/\text{VXV} > 1.083$
- Long XIV/SVXY when $\text{ratio} < \text{SMA}(\text{ratio}, 60)$ and $\text{ratio} < 1$, idem for VXX, where $\text{ratio} = \text{VIX6M} = \text{VIX3M}/\text{VXMT}$.

To trade variance and jump risks around macroeconomic announcements, use long VIX positions and SPX straddles.

The present value of an uncertain stream of cash flows, for an investor with time-separable power utility,

$$\sum_{t \geq 0} \delta^t E \left[\frac{X_t^{1-\gamma}}{1-\gamma} \right]$$

is (Rubinstein’s model, **stochastic discount factor**)

$$\text{PV} = \sum_{t \geq 1} \frac{E[X_t / (1 + r_t^{\text{mkt}})^{\gamma}]}{E[(1 + r_t^{\text{mkt}})^{1-\gamma}]}$$

This can help price options on illiquid assets, such as private equity.

To build an ESG strategy:

- Survey Americans to identify the most important issues (workers, customers, products, environment, community, US jobs, management, shareholders);
- Collect 100 company-level ESG metrics (customer reviews, fines, emissions, etc.);
- Build a portfolio (top half in each industry, market capitalized).

Daily fund flows depend on the Morningstar click rate (raw, trend, change) for some funds, even after controlling for Net asset, Benchmark return, Excess return, Alpha, Market beta, Size beta, Value beta, Momentum beta [but they did not control for multiple testing].

Seasonality. [Several empty talks]

Default models. The **Q-Gaussian** distribution maximizes the Tsallis entropy (the Gaussian maximizes the Shannon entropy); its fatter tails lead to a more realistic Merton distance to default for investment-grade companies.

Bayesian additive regression trees (BART) are forests of regression trees, with a prior on tree structure – monotonicity complicates the definition of the prior and the sampling procedure. Monotonically-constrained BART can help forecast bankruptcies.

The **mvord** package fits models of the form

$$\text{rating} \sim \text{firm} + \text{rater} + \text{time}$$

accounting for correlation between raters and autocorrelation over time.

Graphs. Country modularity, in the graph of non-US companies and directors / analysts / news mentions is higher than sector modularity. Companies sharing directors/analysts/news mentions tend to be more correlated.

Stochastic block models are an alternative to modularity for community detection; they can be generalized to bipartite graphs. Implementation in `dbisbm`.

Text. The `sentometrics` and `quanteda` packages provide lexicon-based sentiment analysis.

To estimate the sentiment (or “tone”) of a document (10k filings) from lists of positive and negative words, one can

- count them;
- weigh them using tf-idf weights;
- weigh them using weights estimated from subsequent market returns.

Abnormal tone is defined by analogy with abnormal returns, as the residuals of

$$\text{tone} \sim \text{market tone} + \text{industry tone}.$$

***Exploring and measuring
non-linear correlations: copulas,
lightspeed transportation and clustering***
G. Marti et al.

Many measures of correlation can be seen as a “distance” between the data copula and some reference copula, e.g., the Fréchet-Hoeffding bounds (comonotonic and counter-monotonic) or the independence copula.

Common divergences (KL, etc.) are not very relevant when dependence is high (given two probability densities, with disjoint supports, they can just tell that the supports are disjoint, not how far apart they are). The **Wasserstein distance** (optimal transport, earth mover’s distance (EMD)) addresses this. Its exact computation is cumbersome, $O(m^3 \log m)$, but its entropic relaxation, the *Sinkhorn distance* (an upper bound of the EMD) can be computed in linear time. The Sinkhorn distance can also be used to compute barycenters of copulas, or to cluster copulas (k -means).

The *target-forget dependence coefficient* of a copula is computed from its Sinkhorn distance to a set of target (e.g., independence) copulas $(C_k^+)_k$, and a set of forget (e.g., Fréchet-Hoeffding) copulas $(C_\ell^-)_\ell$.

$$\text{TFDC}(C) = \frac{\text{Min } d(C_\ell^-, C)}{\text{Min } d(C_\ell^-, C) + \text{Min } d(C_k^+, C)}$$

The *mutual information* $\text{MI}(X, Y)$ is the Kullback-Leibler divergence between the joint distribution (X, Y) and the corresponding independent distribution $p_X p_Y$,

$$\text{MI}(X, Y) = \text{KL}(p_{XY} \| p_X p_Y).$$

***Equitability, mutual information
and the maximal information coefficient***
J.B. Kinney and G.S. Atwal

Mutual information satisfies the *data processing inequality*: $I(X; Z) \leq I(Y; Z)$ if $X \leftrightarrow Y \leftrightarrow Z$ is a

Markov chain (*i.e.*, information is lost, never gained, when transmitted through a noisy channel).

A dependence measure $D(X; Y)$ is *self-equitable* if $D[X; Y] = D[f(X); Y]$ whenever $X \leftrightarrow f(X) \leftrightarrow Y$ is a Markov chain (in particular, if f is bijective).

Mutual information is self-equitable, MIC is not.

***A robust-equitable copula dependence measure
for feature selection***
Y. Chang et al. (2016)

A dependence measure $D(X; Y)$ is *robust equitable* if $D(X; Y) = p$ when (X, Y) has copula $pC + (1 - p)\Pi$, where C is a singular copula and Π the independence copula (self-equitability deals with additive noise rather than mixture noise; both try to be agnostic to the type of relation they detect).

The *robust copula dependence* CD_1 is the L^1 distance between the copula density and the uniform density

$$\text{CD}_\alpha = \iint_{[0,1]^2} |c(u, v) - 1|^\alpha du dv;$$

it can be estimated with a kernel density estimator. It is robust equitable; mutual information and CD_2 are not.

***Copula correlation:
an equitable dependence measure
and extension of Pearson’s correlation***
A.A. Ding and Y. Li

More details on the copula correlation (aka robust copula dependence), with comparisons with other measures of dependence.

***Sinkhorn distances: lightspeed computation
of optimal transportation distances***
M. Cuturi (2013)

The *optimal transport distance* (earth mover’s distance (EMD), Wasserstein distance) between two histograms $r, c \in \Delta_d$, wrt a cost matrix $M \in \mathbf{R}^{d \times d}$, is

$$d(r, c) = \text{Min}_{p \in U(r, c)} \langle P, M \rangle$$

where the transportation polytope $U(r, c)$ is the set of joint probabilities whose margins are r and c ,

$$U(r, c) = \{ P \in \mathbf{R}_+^{d \times d} : P\mathbf{1}_d = r, P'\mathbf{1} = c \}$$

The **Sinkhorn distance** only considers the joint probabilities close to the independence one rc'

$$d_\alpha(r, c) = \text{Min}_{P \in U_\alpha(r, c)} \langle P, M \rangle$$

$$U_\alpha(r, c) = \{ P \in U(r, c) : \text{KL}(P \| rc') \leq \alpha \}.$$

The dual problem (obtained by replacing the hard constraint with a penalty) can be solved efficiently with a fixed point algorithm.

In R: `Barycenter::SinkhornDistance` (buggy).

Fast computation of Wasserstein barycenters

M. Cuturi and A. Doucet (2014)

The Sinkhorn distance (entropic regularization of the Wasserstein distance) makes the computation of Wasserstein barycenters efficient.

The randomized dependence coefficient

D. Lopez-Paz et al.

The Hirschfeld-Gebelein-Rényi **maximum correlation coefficient** is

$$\text{hgr}(X, Y) = \sup_{f, g} \text{Cor}(fX, gY).$$

To estimate it, uniformize X and Y , transform them in several random ways (e.g., $\cos(w'X + b)$) and compute the corresponding *canonical correlation* (i.e., the maximum correlation between a linear combination of the transforms of X and one of those of Y).

Estimating optimal transformations for multiple regression and correlation

L. Breiman and J.H. Friedman (1985)

The *alternating conditional expectation* (ACE) algorithm looks for transformations ϕ, θ of random variables X, Y maximizing the correlation $\text{Cor}(\phi X, \theta Y)$ (not unlike generalized additive models).

While $E[\theta Y - \phi X]^2$ decreases:

$$\phi(X) \leftarrow E[\theta(Y)|X]$$

$$\theta(Y) \leftarrow \frac{E[\phi(X)|Y]}{\|E[\phi(X)|Y]\|}$$

A kernel two-sample test

A. Gretton et al. (2012)

A *reproducing kernel Hilbert space* (RKHS) is Hilbert space of functions $X \rightarrow \mathbf{R}$ (for some unspecified scalar product) such that the evaluation functions

$$\text{ev}_x : \begin{cases} \mathcal{H} & \longrightarrow \mathbf{R} \\ f & \longmapsto f(x) \end{cases}$$

be continuous; they are then (Riesz) of the form $f(x) = \langle f, k(x, \cdot) \rangle$, for some function (“kernel”) k . Conversely, a symmetric positive definite kernel k on X defines a unique RKHS.

For universal RKHS (k continuous and \mathcal{H} dense in \mathcal{C}_b^0 for L^∞), each probability distribution p has a *mean embedding*

$$\mathbb{E}_{x \sim p} [f(x)] = \langle f, \mu_p \rangle,$$

and the **maximum mean discrepancy** (MMD) distance is the distance between those embeddings.

$$\text{MMD}(p, q) = \sup_{\|f\| \leq 1} \mathbb{E}_{x \sim p} f(x) - \mathbb{E}_{y \sim q} f(X) = \|\mu_p - \mu_q\|$$

The plugin estimator of MMD^2 (from the kernel) is biased, but it is easy to obtain an unbiased one, and even

a linear-complexity one

$$\text{MMD}^2 = \frac{1}{m/2} \sum_1^{m/2} h(z_{2i-1}, z_{2i})$$

$$h(z_i, z_j) = k(x_i, x_j) + k(y_1, y_j) - k(x_i, y_j) - k(x_j, y_i);$$

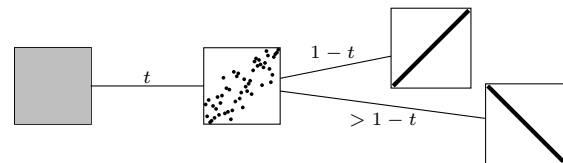
those estimators lead to statistical tests to determine if two samples come from the same distribution.

Optimal copula transport for clustering multivariate time series

G. Marti et al.

Replace d -dimensional time series by their copulas, and use the earth mover’s distance to cluster them.

Define a measure of dependence as the relative distance between the independence copula and one or more target copulas; this also reveals which type of dependence was detected.



A proposal of a methodological framework with experimental guidelines to investigate clustering stability on financial time series

G. Marti et al.

To check the stability of a clustering algorithm for time series, look at how the *adjusted Rand index* (ARI) changes under perturbations:

- Sliding windows of different sizes;
- Odd vs even observations;
- Economic regime (for some regime detection algorithm);
- Whether market returns are in the interquartile range or not;
- Different sampling frequencies;
- Different universes;
- For fixed income: different maturities, or all the maturities together (Hellinger distance between the probability distribution functions $\partial \text{PD} / \partial t$).

On clustering financial time series: a need for distances between dependent random variables

G. Marti et al.

The *hierarchical clustering correlation matrix estimator* is consistent for the *hierarchical correlation block model* (HCBM).

Cluster analysis for portfolio optimization

V. Tola et al.

Original paper on the ultrametric correlation matrix from hierarchical clustering.

Toward a generic representation of random variables for machine learning
G. Marti et al.

The L^2 distance between random variables $d^2(X, Y) = E(X - Y)^2$ mixes dependence and distribution information: it is often unsuitable to cluster iid sequences. Instead, use a convex combination of the *correlation distance* (of the copula (X, Y) , for the dependence side) and of the *Hellinger distance* (between the marginals, for the distribution side).

Copula-based kernel dependency measures
B. Póczos et al. (2012)

The **maximum mean discrepancy** (MMD) between two probability distributions P, Q is

$$M = \sup_{f \in \mathcal{F}} E_{X \sim P}[f(X)] - E_{Y \sim Q}[f(Y)]$$

where \mathcal{F} is the unit ball in a universal reproducing kernel Hilbert space (RKHS) (universal means dense in \mathcal{C}_b^0 , e.g., with the Gaussian or Laplace kernels); M^2 can be estimated as

$$E_{X, X' \sim P}[k(X, X')] - 2 E_{\substack{X \sim P \\ Y \sim Q}}[k(X, Y)] + E_{Y, Y' \sim Q}[k(Y, Y')].$$

The *dependence* I^2 between random variables X_1, \dots, X_d is the MMD distance between their joint copula and the uniform distribution; it can be estimated as

$$\frac{1}{m(m-1)} \sum_{i,j} k(Z_i, Z_j) + k(U_i, U_j) - k(Z_i, U_j) - k(U_i, Z_j)$$

or

$$\frac{1}{m^2} \sum_{ij} k(Z_i, Z_j) + \frac{1}{n^2} \sum k(U_i, U_j) - \frac{2}{mn} \sum k(Z_i, U_j)$$

where $U_i \sim U(0, 1)^d$ and Z_i is the empirical copula of the data X_i .

This dependence measure can be used for *feature selection*:

Find	$S \subset \llbracket 1, d \rrbracket$
To minimize	$ S $
And maximize	$I(X_S \cup Y)$

Optimal transport vs Fisher-Rao distance between copulas for clustering multivariate time series
G. Marti et al.

The Fisher-Rao metric and its tractable approximations (Kullback-Leibler and other divergences – computing geodesics requires solving ODEs) are unsuitable to compute distances between copulas, especially when the dependence is high: the Gaussian copula with correlation 0.99 appears closer to that with correlation 0.50 than to that with correlation 0.9999 – the monotonic copula, $\rho = 1$, does not have a density and is

not on the manifold – it is at infinity. The Wasserstein distance is preferable.

Cluster multivariate time series by replacing them with their copulas.

Autoregressive convolutional neural networks for asynchronous time series
M. Bińkowski et al. (2018)

Irregular time series can be made regular by considering $y_n = (x_{t_n}, t_n - t_{n-1})$; for multivariate asynchronous time series (e.g., quotes for the same asset from different market makers), put all the values together and add a 1-hot encoding of the source.

The significance-offset CNN (SOCNN)

$$x_n = \sum_{m=1}^M W_{.,m} \odot [x_{n-m} + \varepsilon(x_{n-m})] \odot \sigma S_{.,m}(x_{n-1}, \dots, x_{n-m})$$

where ε is dense, shallow, S convolutional and σ the softmax, combines autoregression (for $\sigma = 1$, $\varepsilon = 0$, the recent observations are remembered and used as is) and gated RNN (older observations are summarized, in a nonlinear way).

Hausdorff clustering of financial time series
N. Basalto et al. (2005)

The **Hausdorff distance** gives an alternative to single and complete linkage clustering.

$$\begin{aligned} d(A, B) &= \inf_{a \in A} \sup_{b \in B} d(a, b) \\ d(A, B) &= \sup_{a \in A} \sup_{b \in B} d(a, b) \\ d(A, B) &= \text{Max} \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \right. \\ &\quad \left. \sup_{b \in B} \inf_{a \in A} d(a, b) \right\} \end{aligned}$$

One could also switch Sup and Inf (minimax clustering).

Hierarchical clustering with prototypes via minimax linkage
J. Bien and R. Tibshirani (2011)

The *minimax radius* of a cluster C is

$$r(C) = \text{Min}_{x \in C} \text{Max}_{y \in C} d(x, y);$$

the minimizer is a *prototype* of C . *Minimax linkage* hierarchical clustering uses $d(A, B) = r(A \cup B)$.

HCMapper: an interactive visualization tool to compare partition-based flat clustering extracted from pairs of dendrograms
G. Marti et al.

Compare the partitions implied by two dendrograms with a *Sankey diagram* (D3) going from the root of the first one, to the leaves, to the root of the second one, displaying the size of the intersections.

***NetGAN: generating graphs
via random walks***
A. Bojchevski et al. (2018)

Consider random walks of length T on a graph as sequences and train a Wasserstein GAN on them: stop training when the (validation set) link prediction performance drops, or when the edge overlap between the generated graph and the data reaches a user-specified threshold.

***Geometry score: a method for comparing
generative adversarial networks***
V. Khrulkov and I. Oseledets (2018)

To assess the quality of a GAN, compare the data manifold and the generated manifold using:

- The Wasserstein distance between the birth-death diagrams;
- The distance between the persistence landscapes;
- The proportion of time $\beta_1 = i$ [they do not seem to notice this discards the persistence].

***Unsupervised representation learning with deep
convolutional generative adversarial networks***
A. Radford et al. (2016)

To train deep convolutional GANs (DCGAN),

- Replace the pooling layers with strided convolutions;
- Use batch normalization;
- Remove the fully-connected layers;
- Use ReLU in the generator (and tanh for the output) and leaky ReLU in the discriminator.

***StackGAN:
text to photo-realistic image synthesis
with stacked generative adversarial networks***
H. Zhang et al.

To generate high-resolution images from text descriptions using GANs, stack two of them. If there is not enough training text data, add noise to the sentence embedding.

***AlignedReID: surpassing human-level
performance in person re-identification***
X. Zhang et al.

To detect if two pictures (from surveillance cameras, to track visitors in a mall) are of the same person, compute both global and local features, where the local features are from horizontal bands in the image, and are aligned between the two pictures.

***Re-ranking person re-identification
with k -reciprocal encoding***
Z. Zhong et al.

Retrieval systems return a ranking, but do not always use all the information available: in particular, true positives are more similar among themselves. Replace the original distance with a linear combination of the

original distance and the (weighted) Jaccard distance between the set of k -reciprocal neighbours (x and y are k -reciprocal neighbours if each is among the k nearest neighbours of the other), which can be computed on binary vectors, with the 1s replaced with e^{-d} , where d is the distance to the probe.

Deep mutual learning
Y. Zhang et al.

Model distillation first trains a deep neural net on data, then a shallow one to mimic it.

Deep mutual learning uses an ensemble of shallow networks, trained with two losses, a supervised one, and a mimicry one.

Deep learning
A. Ng (Coursera, 2017)

1. With big data, the dev and test sets can be much smaller: 10,000 observations is more than enough, but that can be less than 1% of the data.

High bias is bad performance on the training data; high variance is bad performance on the test set.

Early stopping is a crutch to prevent overfitting: it mixes two objectives, reducing bias (improving in-sample performance) and reducing variance (avoiding overfitting) – other regularization methods (penalties, dropout, etc.) give more control on the bias-variance tradeoff.

Rather than opposing bias and variance, we should oppose “avoidable bias” (or “Bayes error”: the difference between the error and the lowest possible error, estimated, e.g., with the human-level error) and variance.

If the training and dev sets do not come from the same distribution, the difference in performance can come either from an excessive variance or the difference itself. To disentangle those effects, use a “training-dev set” – another dev set, carved out of the training set, to measure variance (i.e., whether the model generalizes to unseen data from the same distribution).

2. Auto-encoders can be used for anomaly detection: for anomalies (inputs from a never-seen class), the reconstruction error is higher.

End-to-end learning needs more data (10 to 100 times more) than traditional multi-step pipelines. Do not use if you have more data for the individual steps than for the whole pipeline.

ResNets can easily learn the identity function (while deep “plain” nets cannot): that is why we can make them as deep as possible. We can also increase their depth by adding new residual layers, initialized as the identity, in the middle.

3. For images, prefer Inception-v4 – VGG16 and VGG19 are larger and have a slightly lower accuracy.

In a CNN, the size progressively decreases, but the number of channels increases.

Average pooling is rarely used, except sometimes at the end, to reduce the size of the final result, e.g., $7 \times 7 \times 1000 \rightarrow 1 \times 1 \times 1000$. Max-pooling is more common.

1×1 convolution (aka “network in network”) is actually a fully-connected network acting on each pixel (the same network for each pixel). (It should not be written 1×1 but $1 \times 1 \times n$, where n is the number of input channels.) There are usually several of them: they are specified by a $1 \times 1 \times n \times m$ tensor, where m is the number of filters, i.e., the number of output channels. Those *bottleneck layers* (1×1 convolutions) are often used to reduce the number of channels before a convolution, to reduce the number of operations (and parameters). For instance, the inception units have a 1×1 convolution before their 5×5 convolutions.

If you do not know if a 1×1 , 3×3 , 5×5 convolution or a maxpooling operation is better, you can do them all, and concatenate the results (inception module: in Keras, `merge([a,b,c,d],mode='concat')`). To reduce the computations, you may want to first reduce the number of layers with a 1×1 convolution. Inception modules also have a maxpool operation (which does not change the number of channels): the 1×1 convolution comes after. The inception network also has a few side outputs, trying to forecast the same output – this helps learning and has a regularizing effect.

To recognize faces with neural nets, feed the neural network pairs of images (faces, extracted from wider images) and have it predict if the two images are of the same person.

To recognize faces, learn a dissimilarity function $d(\text{img}_1, \text{img}_2)$ (siamese network – deepface).

The *triplet loss* (facenet) tries to enforce $d(x_i, x_j) - d(x_i, x_k) + \alpha \leq 0$ (with a margin, α , as for SVMs), using the positive part of the lhs as loss. Choose triplets that are hard to train on; train on 1m to 100m images.

For object localization, have the convnet output: a boolean indicating if an object is present, a 1-hot encoding of the object class, the coordinates of the center of the object bounding box, its width and height (if no object is present, the loss function ignores the predicted class and bbox).

For object detection (there can be several objects), use a sliding window. It can be implemented as a single convolutional neural net (keep the same weights, just change the input size – and rewrite the FC layer as a convolutional one, outputting a $1 \times 1 \times 1$ tensor).

YOLO is similar, with no overlap between the windows, and a localization network (not just a classification one).

For object detection, the ratio of intersection over union (IoU) is a possible loss function.

To avoid detecting the same object in nearby locations, keep the object with the largest object presence probability (non-max suppression), discard the detected ob-

jects with a large overlap (it is the same object), and iterate with the remaining (different) objects; do this independently for each class.

To deal with overlapping objects, predict the presence of several objects with different bbox aspect ratios (anchor boxes): if there are k aspect ratios, the output is k times larger.

Do not run the object detection on the whole region but only on “promising” regions (R-CNN) identified as “blobs” by some segmentation algorithm (classical or CNN – “faster R-CNN”).

To do well in ML competitions use ensembling, and multi-crop at test time (useless in production).

4. The *BLEU* (bilingual evaluation understudy) score, for machine translation, counts the number of 1-, 2- and 3-grams in the computer-generated sentence that are also present in one of the reference translations; it also includes a brevity penalty.

When generating a sequence with an RNN, one needs to select words one by one. The greedy algorithm is suboptimal – it will favour common words. *Beam search* takes the b best words at each step. Use $b = 10$ in production, $b = 100$ to 10,000 for research.

Beam search is biased towards shorter sentences: use the average instead of the sum of the conditional log-probabilities of the successive words (length normalization).

To analyze the error, compare the score of a correct (human-generated) sentence with that of the sentence actually generated, to check if the problem comes from beam search (increase b) or the RNN.

Non-convex multi-objective optimization P.M. Pardalos et al. (2017)

A multiobjective optimization problem

Find	x
To minimize	$f_1(x), \dots, f_n(x)$
Such that	$g(x) \geq 0$

asks for the *Pareto set* (non-dominated solutions – the Pareto *front* is the set of corresponding values). For convex problems, local Pareto optimality implies global optimality, and the solutions are the minimizers of $\sum \alpha_i f_i$ for some α , for various values of α .

Scalarization replaces the objective functions with a single function (this need not give all solutions, and may also produce non-solutions):

- Weighted sum: $\sum w_i f_i(x)$ (for convex problems);
- Weighted sum after transformation into a convex problem;
- Tchebychev: minimize $\text{Max}_i w_i (f_i(x) - u_i)$, where u is a utopian point, $u_i < \text{Min } f_i$; the problem can be reformulated as

Minimize t such that $w_i (f_i(x) - u_i) \leq t$,
 x, t

and one can add a penalty for the Manhattan distance to the utopian point;

- Achievement scalarization: idem, but with an arbitrary point instead of a utopian one;
- Minimize only one objective and use the others as constraints; the constraints may combine several objective functions (normalized normal constraint, normal boundary intersection);
- Pascoletti and Serafini Scalarization

$$\underset{t,x}{\text{Minimize}} t \text{ such that } u + tv - f(x) \geq 0$$

where u and v are arbitrary vectors.

The parameters in those methods can rarely be chosen beforehand: select them adaptively – for instance, with two objectives, and weights (w_1, w_2) as parameters, start with uniformly-distributed weights, and select which interval to refine/subdivide/bisect.

The bi-objective shortest path problem on a graph (with two weightings) can be reduced to the 0-1 knapsack problem: it is NP-complete. The size of an ε -solution is polynomial, but it may be exponential in the number of objectives.

Global optimization methods include Bayesian optimization (under other names: P-algorithm, π -algorithm, efficient global optimization (EGO), kriging) and branch-and-bound (under a Lipschitz assumption, one can find lower bounds of the objectives in each region, and progressively refine promising regions – probabilistic variants use extreme value theory to estimate the minimum in each region).

Generative adversarial nets

I.J. Goodfellow et al.

Generative models can be estimated via an adversarial process, corresponding to a minimax two-player game,

$$\underset{G}{\text{Min}} \underset{D}{\text{Max}} \underset{x \sim \text{Data}}{\mathbb{E}} [\log D(x)] + \underset{z \sim \text{noise}}{\mathbb{E}} [\log(1 - D(G(z)))].$$

Replacing $\log(1 - DGz)$ with $-\log(DGz)$ helps learning early on.

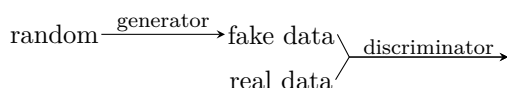
The objective function is related to the Jensen-Shannon metric:

$$\underset{x \sim \text{Data}}{\mathbb{E}} [\log D(x)] - \underset{z \sim \text{noise}}{\mathbb{E}} [\log D(G(z))] \leq 2\text{JS}(\text{Data}, G(\text{noise})) - 2\log 2$$

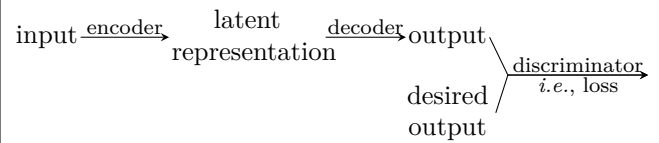
Image-to-image translation with conditional adversarial networks

P. Isola et al.

The adversarial approach of GANs



can be merged with autoencoders (with skip-connections) for image translation.



Wasserstein GAN

M. Arjovsky et al.

The Wasserstein distance is more relevant than the Kullback-Leibler divergence for distributions supported by low-dimensional manifolds: KL just informs us that the supports are disjoint, while Wasserstein also tells us how far apart they are.

$$\text{TV}(P, Q) = \sup_A |P[A] - Q[A]|$$

$$\text{KL}(P, Q) = \int p \log \frac{p}{q}$$

$$\text{JS}(P, Q) = \text{KL}\left(P, \frac{P+Q}{2}\right) + \text{KL}\left(Q, \frac{P+Q}{2}\right)$$

$$W(P, Q) = \sup_{\gamma} \underset{\substack{\text{pr}_1 \gamma = P \\ \text{pr}_2 \gamma = Q}}{\mathbb{E}} [\|x - y\|]$$

If X and Z are random variables, and $g_\theta(z)$ is continuous in θ , then $W(g_\theta(Z), X)$ is continuous in θ ; if g is locally Lipschitz, then W is differentiable almost everywhere.

The Wasserstein distance is weaker (it has more converging sequences): on compact spaces,

$$\begin{array}{ccccc} \text{KL} & \longrightarrow & \text{JS} & \longrightarrow & W \\ & & \updownarrow & & \updownarrow \\ & & \text{TV} & & \mathcal{D}. \end{array}$$

The Kantorovich-Rubinstein duality

$$W(P, Q) = \sup_{\|f\|_{\text{Lipschitz}} \leq 1} \underset{x \sim P}{\mathbb{E}} [f(x)] - \underset{x \sim Q}{\mathbb{E}} [f(x)]$$

leads to a GAN-like optimization problem

$$\underset{G}{\text{Min}} \underset{D}{\text{Max}} \underset{x \sim \text{Data}}{\mathbb{E}} [D(x)] - \underset{z \sim \text{Noise}}{\mathbb{E}} [D(G(z))]$$

where the discriminator D should be 1-Lipschitz (otherwise, the objective is unbounded); this can be enforced, for instance, by using neural nets with clipped weights.

Contrary to traditional GANs, the gradient remains meaningful: the discriminator can be trained to optimality, and the corresponding Wasserstein distance can help monitor convergence. WGANs do not seem to suffer from mode collapse, but they are unstable when trained with momentum – prefer RMSProp.

The same idea can be used for *integral probability metrics* (IPM):

- Bounded functions give the total variation (TV) distance and energy-based GANs (EBGAN);
- The L^∞ ball in a reproducible kernel Hilbert space $\mathcal{H} = \{k(x, \cdot) : x \in \mathcal{X}\}$ for some kernel $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R}$ gives the maximum mean discrepancy (MMD) metric and generative moment matching networks (GMMN).

Improved training of Wasserstein GANs I. Gulrajani et al.

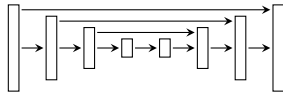
Replace the weight clipping with a penalty on the gradient, ensuring its norm is close to 1 (rather than below 1) on straight lines between the data and the generated data,

$$\lambda \mathbb{E}_x (\|\nabla D|_x\| - 1)^2, \quad \lambda = 10.$$

Fully convolutional networks for semantic segmentation J. Long et al.

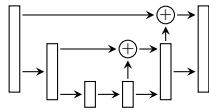
To turn a classification network into a segmentation one,

- Start with a classifier;
- Replace the fully-connected layers with 1×1 convolutions;
- Add up-sampling and skip-layer connections.



Time series modeling with undecimated fully convolutional neural networks R. Mittelman

Remove the max-pooling and upsampling layers from the FCN, but use “upsampled filters” instead (*i.e.*, replace an FCN filter F with $F \otimes \mathbf{1}_{k \times k}$: it is larger, but has the same number of parameters).

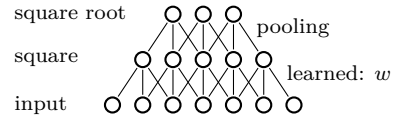


Tiled convolutional neural networks Q.V. Le et al. (2010)

A topological independent component analysis (TICA) network is a 2-layer network whose first layer weights are local and learned, whose second layer is a pooling layer, with square and square root activations respectively, loss function

$$\sum_{t \in \text{Training}} \sum_{i \in \text{Output}} \sqrt{\sum_{k \in \text{Hidden}} \left(\sum_{j \in \text{Input}} w_{kj} x_{jt} \right)^2}$$

and an orthogonality constraint $WW' = I$.



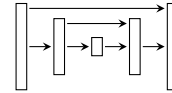
The hidden layer is a sparse representation of the data.

One can obtain an over-complete representation of the data (hidden layer larger than the input) by using local receptive fields (each hidden node only depends on a few input coordinates) and imposing orthogonality only for hidden nodes sharing the same input.

In a *tiled convolutional neural net*, hidden units k steps away share their weights (for a CNN, $k = 1$). They can be pre-trained with TICA, and capture invariances beyond translation (rotation, etc.).

A closed-form solution to photorealistic image stylization Y. Li et al.

For style transfer, start with an autoencoder with unpooling layers



and VGG19 as the encoding part, apply a *whitening and colouring transform* (WCT) to the VGG features

$$\begin{aligned} H_c &\mapsto H = P_s P_c H_c \\ P_c &= E_c \Lambda_c^{-1/2} E_c \\ P_s &= E_s \Lambda_s^{+1/2} E_s \end{aligned}$$

where Λ_c , Λ_s are the eigenvalues of $H_c H_c'$ and $H_s H_s'$ and E_c , E_s their eigenvectors, which makes the correlations of the transformed features HH' match the style ones, $H_s H_s'$, and smooth the result

$$\text{Argmin}_r \sum_{ij} w_{ij} \left\| \frac{r_i}{\sqrt{w'_i \mathbf{1}}} - \frac{r_j}{\sqrt{w'_j \mathbf{1}}} \right\|^2 + \lambda \sum_i \|r_i - y_i\|^2$$

where y is the WCT image, r the final image, and w_{ij} the *matting affinity* of pixels i and j (based on the means and variances of the pixels in a local window) – it is a quadratic optimization problem, with a closed form solution.

Metric learning with adaptive density discrimination O. Rippel et al. (2016)

To help separate the classes in distance metric learning (DML), do not only use a triplet loss, model the distribution of each class in the new space and use a *magnet loss* to pull them apart.

Pixel recurrent neural networks
A. van den Oord et al. (2016)

Recurrent neural networks can be generalized to 2-dimensional structures – for instance, a bidiagonal bi-LSTM combines two grids of LSTMs, one going right and down, the other left and down.

Using natural language processing techniques for stock return predictions
M.L. Chew et al. (2017)

Many studies use a vector embedding and a CNN to forecast post-event price movements [one could also use a bidirectional LSTM with attention, instead of a CNN].

- Use the Stanford NER tool in NLTK to identify company names in news headlines;
- Heuristically map them to tickers (for each company, build a list of possible names, but remove frequent words);
- Infer the structure of the sentence with `textacy` (Spacy) of StanfordPOSTagger;
- Compute Glove embeddings of verb-object pairs;
- Cluster them into 100 event types.

An interpretable and sparse neural network model for nonlinear Granger causality discovery
A. Tank et al. (2017)

Adding a group lasso (or a hierarchical lasso) penalty to the first layer weights of a neural net $x_{i,t} \sim x_{\cdot, < t}$ can help detect (nonlinear) causality and select the number of lags.

Clustering correlated, sparse data streams to estimate a localized housing price
Y. Ren et al.

House prices can be modeled at the census level.

$$\begin{aligned} g_t &: \text{global market trend} \\ x_{it} &: \text{value of census tract } i \\ y_{it\ell} &: \text{value of house } \ell \\ U_\ell &: \text{hedonics (house features)} \\ x_{it} &= g_t + a_i(x_{i,t-1} - g_{t-1}) + \varepsilon_{it} \\ y_{it\ell} &= x_{it} + f_i(U_\ell) + \eta_{it\ell} \\ \varepsilon &\sim N(0, \Sigma), \Sigma \text{ block diagonal} \end{aligned}$$

A unified framework for missing data and cold start prediction for time series data
C. Xie et al.

Matrix factorization (for time series) can be combined with matrix completion (from collaborative filtering):

$$Y_i = LR_i + HU\phi_i + b + \varepsilon_i$$

where R_i are latent time series, ϕ_i features, $U\phi_i$ dimension-reduced features.

Quasi-recurrent neural networks
J. Bradbury et al. (2017)

Model sequential data with *masked convolutions* * (*i.e.*, convolutions using only past data) outputting LSTM elements

$$\begin{aligned} Z &= \tanh(W_z * X) \\ F &= \sigma(W_f * X) \\ O &= \sigma(W_o * X) \end{aligned}$$

combined as $h_t = f_t \odot h_{t-1} + (1 - f_t) \odot z_t$ or

$$\begin{aligned} c_t &= f_t \odot c_{t-1} + (1 - f_t) \odot z_t \\ h_t &= o_t \odot c_t. \end{aligned}$$

Add skip connections (dense convolution), dropout (zoneout), attention.

Lie access neural Turing machine
G. Yang (2016)

In addition to random access, allow translations from one memory cell to another, or the action of some other Lie group (\mathbf{R}^2 , $\text{SO}(2)$, $\mathbf{R}^2 \rtimes \text{SO}(2)$ acting on \mathbf{R}^2).

Learning to optimize
K. Li and J. Malik

Optimization algorithms can be learnt as reinforcement learning policies, with *guided policy search*.

Extrapolation and learning equations
G. Martius and C.H. Lampert (2016)

To learn small, interpretable analytical expressions, amenable to extrapolation, use a neural network whose activation functions are the desired building blocks (the identity, sin, cos, sigmoid – one could also add products and divisions) with a sparsity constraint.

Adversarially learned inference
V. Dumoulin et al. (2017)

Combine generative adversarial networks (GAN, *i.e.*, generator and discriminator) and variational autoencoders (VAE, *i.e.*, inference and generation):

- Encoder, from real data x to latent representation z : $p(x, z) = p(x)p(z|x)$;
- Decoder (generator), from a latent representation z to simulated data x : $p(x, z) = p(z)p(x|z)$, with $p(z) \sim \text{Gaussian}$;
- Discriminator to guess if (z, x) is real or simulated.

Learning to discover sparse graphical models
E. Belilovsky (2017)

Incorporating a prior (e.g., small world) into the graphical lasso can be difficult: instead, use a neural net (CNN, for an arbitrary order on the nodes) to recover the graph from the sample correlation matrix, trained on synthetic data generated by the desired prior.

Deep learning with sets and point clouds
S. Racanbakhsh et al. (2017)

If the input of a layer is a set (if its order is irrelevant). use equivariant transformations, e.g.,

$$\begin{aligned} f(\mathbf{x}) &= \sigma(\lambda \mathbf{x} + \gamma(\mathbf{1}'\mathbf{x})\mathbf{1}) \\ f(\mathbf{x}) &= \sigma(\lambda \mathbf{x} - \gamma(\text{Max } \mathbf{x})\mathbf{1}) \\ f(\mathbf{x}) &= \sigma(\beta + \mathbf{x}\Gamma - \mathbf{1}(\text{Max } \mathbf{x})\Gamma) \end{aligned}$$

Applications include MNIST digit summation (the order of the digits is irrelevant) and point cloud classification, or situations with an existing clustering.

Normalizing the normalizers: comparing and extending network normalization
M. Ren et al. (2017)

Batch normalization, layer normalization (for RNNs) and *divisive normalization* (normalize neurons' activities by their neighbours') are the same operation along different dimensions of a tensor.

Regularizing CNNs with locally constrained decorrelation
P. Rodríguez et al. (2017)

To ensure the features learnt are as diverse as possible, add a penalty $\text{Cor}(\theta_i, \theta_j)^2$ if i and j are neurons in the same layer. To preserve negative correlations, replace Cor^2 with $\text{softplus}(\text{Cor})$ or $\text{softplus}(\lambda(\text{Cor} - 1))$.

Orthogonal weights are also sometimes used (with Xavier scaling) to initialize neural nets. Multibias neural nets have neurons sharing weights but with different biases.

Hierarchical multiscale recurrent neural networks
J. Chung et al. (2017)

LSTM with sparse updates, triggered by a boundary detector.

Geometry of polysemy
J. Mu et al.

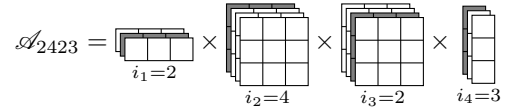
Using traditional word embeddings, represent a context (10 words before and after the target word) as a cloud of (20) points. The span of the first $n = 4$ principal components form a subspace, *i.e.*, a point in the Grassmanian $G_{4,300}$. The different senses of a word can be recovered by clustering (the k -means algorithm can be generalized to the Grassmanian) its contexts; each cluster corresponds to a line – a point in $G_{1,300}$.

Exponential machines
A. Novikov et al. (2017)

A d -dimensional tensor \mathcal{A} has **tensor train** (TT) rank at most r if its elements can be written

$$\mathcal{A}_{i_1 \dots i_d} = G_1[i_1] \cdots G_d[i_d]$$

where $G_1[i_1]$ is $1 \times r$, $G_k[i_k]$, for $1 < k < d$, is $r \times r$, and $G_d[i_d]$ is $r \times 1$. Operations on tensor trains are efficient and return other tensor trains.



The linear model

$$\begin{aligned} \text{find} \quad & w, b \\ \text{To minimize} \quad & \sum \ell(\langle x, w \rangle + b, y) + \lambda \|w\|_2^2 \end{aligned}$$

can be generalized to account for interactions

$$\begin{aligned} \text{Find} \quad & \mathcal{W} \\ \text{To minimize} \quad & \sum \ell(\langle \mathcal{X}, \mathcal{W} \rangle, y) + \lambda \|\mathcal{W}\|_2^1 \\ \text{Such that} \quad & \text{TT-rank } \mathcal{W} = r \\ \text{Where} \quad & \mathcal{X}_{i_1 \dots i_d} = \prod_k x_k^{i_k} \end{aligned}$$

and estimated with Riemannian optimization (tensor trains of fixed rank form a Riemannian manifold).

Decoupled neural interfaces using synthetic gradients
M. Jaderberg et al.

To avoid backpropagating the gradients all the way down the computation graph, use neural networks to forecast them from local information.

Why and when can deep (but not shallow) networks avoid the curse of dimensionality: a review
T. Poggio et al. (2017)

Convolutional neural networks (CNNs) work well, not because of weight sharing, but because they are hierarchical compositions of *local* (*i.e.*, low-dimensional) functions.

Improved variational inference with inverse autoregressive flow
D.P. Kingma et al. (2016)

Variational inference looks for a simple distribution q approximating the posterior distribution p minimizing $D_{KL}(q||p)$. *Normalizing flows* allow more flexibility through reparametrization $z_0 \sim q$, $z_t = f_t(z_{t-1})$, where the f_t 's combine a linear transformation and a nonlinearity.

Convolutional neural networks on graphs with fast localized spectral filtering
M. Defferrard et al. (2016)

Convolutional neural nets can be generalized from regular grids to arbitrary networks:

- Consider the Laplacian $L = D - W$ or the normalized Laplacian $L = I - D^{-1/2} W D^{-1/2}$;
- Diagonalize it, $L = U \Lambda U'$;
- Define the Fourier transform as $x \mapsto U'x$ and the convolution as $x * y = U((U'x) \odot (U'y))$;

- Consider filters of the form $x \mapsto g_\theta(L)x$, where $g_\theta(L) = \sum \theta_k L^k$ or $g_\theta(L) = \sum \theta_k T_k(L)$ (Chebychev polynomials).

**Seven neurons memorizing sequences
of alphabetic images
via spike-timing dependent plasticity**

T. Osogami and M. Otsuka

A **dynamic Boltzman machine** (DyBM) is a Boltzman machine unrolled through time.

**Learning dynamic Boltzman machines
with spike-timing dependent plasticity**
T. Osogami and M. Otsuka (2015)

Although a DyBM can be interpreted as an RNN with memory, it can be learnt using only local information (no vanishing gradient problem).

**Nonlinear dynamic Boltzman machines
for time series prediction**

S. Dasgupta and T. Osogami (2016)

A **Gaussian Boltzman machine** is a Boltzman machine (BM) in which the Bernoulli distributions

$$E_j = b_j + \sum_{i \neq j} w_{ij} x_i \quad P[X_j = 1] \propto e^{-E_j}$$

have been replaced with Gaussian distributions

$$\mu_j = b_j + \sum_{i \neq j} w_{ij} x_i \quad X_j \sim N(\mu_j, \sigma_j).$$

A Gaussian dynamic Boltzman machine (DyBM) is an unfolded Gaussian BM. It can be interpreted as a VAR model, with eligibility traces (exponentially weighted moving averages) as additional variables. The input of a Gaussian DyBM can be preprocessed by an echo state RNN (*i.e.*, an RNN with random, rather than learnt, weights).

**Dynamic Boltzman machines
for second order moments
and generalized Gaussian distribution**

R. Raymond et al. (2017)

Gaussian dynamic Boltzman machines can be generalized, by making the variance time-dependent, as in a GARCH model (but using eligibility traces); and/or by replacing the Gaussian distribution, *e.g.*, with $p(x) \propto \exp[-\beta|x - \mu|^\rho]$.

**Training deep and recurrent networks
with Hessian-free optimization**

J. Martens and I. Sutskever (2015)

Hessian-free (HF) optimization (or **truncated Newton**) approximates the objective function with a quadratic

$$h(\theta + \delta) \approx h(\theta) + G'\delta + \frac{1}{2}\delta'H\delta$$

but computes the update $\theta \leftarrow \theta - H^{-1}G$ using a few *conjugate gradient* (CG) steps: there is no need to invert the curvature matrix, or even to compute it.

- Hessian-vector multiplications are directional derivatives of the gradient: they can be computed with forward differentiation.
- Replace the Hessian with the *generalized Gauss-Newton matrix* (GGN), which assumes $f'' = 0$, (*i.e.*, replaces f with its first order approximation); for scalar functions:

$$\begin{aligned} h(\theta) &= L(f(\theta)) \\ h'(\theta) &= f'(\theta)L'(f(\theta)) \\ h''(\theta) &= f''(\theta)L'(f(\theta)) + f'(\theta)L''(f(\theta))f(\theta) \\ &\approx f'(\theta)L''(f(\theta))f(\theta); \end{aligned}$$

for supervised learning:

$$\begin{aligned} h(\theta) &= \frac{1}{|S|} \sum_{(x,y) \in S} L(y, f(x, \theta)) \\ h''(\theta) &\approx \frac{1}{|S|} \sum_{(x,y) \in S} J'H_L J, \quad J = \nabla_z L, \quad H_L = \nabla_{zz} L; \end{aligned}$$

- Use *Tikhonov damping* (L^2 regularization) and make it scale-sensitive by using the diagonal of the Hessian (Levenberg-Marquardt);
- Use *structural damping*, *i.e.*, a penalty, not just to changes in the parameters, but to some intermediate quantities (*e.g.*, hidden activations – just add them to the output of the network and feed them to the loss function);
- Increase or decrease the scale of the penalty by looking at the ratio $\Delta\text{objective}/\Delta\text{penalty}$ (Levenberg-Marquardt heuristic);
- Instead of a penalty, use a *trust region*;
- Use early CG stopping;
- As a last resort, use a line search;
- Initialize CG with the previous direction (slightly scaled down) rather than 0;
- Use a preconditioner: diagonal preconditioners are effective with deep neural nets, because the scale of the gradients can grow/shrink exponentially between layers – but less so for RNNs, because the weights are reused. You may want to blend the diagonal matrix with the identity, $P = (\text{diag}(d) + \kappa I)^\xi$. There is no efficient algorithm to compute the diagonal of the Hessian: use the diagonal of the *empirical Fisher information*

$$\bar{F} = \frac{1}{|S|} \sum_{(x,y) \in S} \nabla L(y, f(x, \theta)) \nabla L(y, f(x, \theta))'$$

$$d = \text{diag} \bar{F} = \frac{1}{|S|} \sum \text{sq} \nabla L(y, f(x, \theta))$$

sq = elementwise square

(but it is biased) or, for an unbiased estimator,

$$d = \mathbb{E}_{v \sim N(0,1)} [\text{sq} J' H_L^{1/2} v]$$

- If possible, estimate the gradient and the Hessian on the same (large) minibatch.

***Sparse portfolios
for high-dimensional financial index tracking***
K. Benidis et al. (2017)

Passive investors try to reproduce the performance of some reference index with a simpler portfolio: fewer and more liquid assets. This can be formulated as an optimization problem:

Find	w	Portfolio weights
To minimize	$\text{TE}[w, b]$	Tracking error
Such that	$w \geq 0, w'1 = 1$	

For a sparse portfolio, one can add an L^0 penalty, but the resulting problem is too difficult to be solved exactly – there are heuristics, but they do not provide good guarantees on the quality of the results. To make it easier to solve, one can relax the constraint. The usual L^1 relaxation does not work here: the L^1 norm is already constrained to be 1: $\|w\|_1 = w'1 = 1$. Instead, one can use a finer relaxation:

$$\rho_{p,\gamma}(\mathbf{w}) = \frac{\log(1 + |\mathbf{w}|/p)}{\log(1 + \gamma/p)};$$

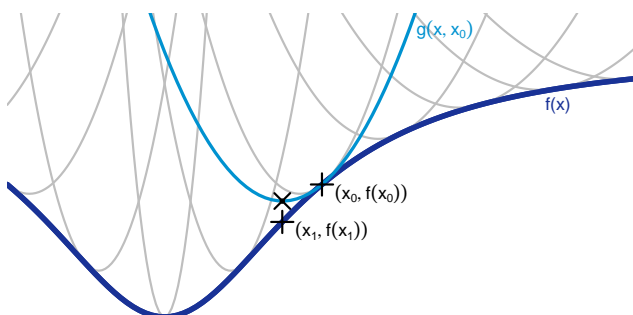
it is not convex, but the optimization problem can be solved by *majorization-minimization* (MM), which reduces non-convex optimization problems to sequences of convex optimization problems. Those optimization problems can be reformulated (again using MM) into a sequence of quadratic programs with a closed form solution – we are left with a simple and very fast iterative algorithm, which no longer requires a solver.

The use of MM to deal with cardinality constraints is more general: for instance, adding a minimum weight constraint (either the weight is zero, or it is above a threshold) leads to a similar algorithm. One can also use other tracking errors, e.g., only pay attention to the downside risk.

To find the minimum of a function f , **majorization-minimization** (MM) looks for a simpler function g such that

$$\begin{aligned} \forall x, y \quad & g(x, y) \geq f(x) \\ \forall y \quad & g(y, y) = g(y) \end{aligned}$$

and iterates $x_{n+1} \leftarrow \text{Argmin}_x g(x, x_n)$.



***Successive convex optimization methods
for risk parity portfolio design***
Y. Feng and D.P Palomar (2015)

Investors usually have to choose between risk parity portfolios and Markowitz-efficient portfolios, but it is possible to combine both loss functions into the same objective, yielding a portfolio close to risk parity, with high returns, with low risk.

***Regularized robust estimation
of mean and covariance matrix
under heavy-tailed distributions***
Y. Sun et al. (2014)

Estimate variance matrices, in the presence of outliers, fat tails and/or insufficient data, with M-estimators (several of them) and a majorization-minimization algorithm.

***Minimal criterion coevolution:
a new approach to open-ended search***
J.C. Brant and K.O. Stanley (2017)

Open-ended evolution (non-objective search) forgoes the objective function and only uses a minimum criterion, which should be satisfied for reproduction. The idea extends to coevolution, e.g., with mazes and maze-solving neural nets, the criteria being that each maze is solved by at least one neural net and each neural net solves at least one maze.

***Stochastic dual coordinate ascent methods
for regularized loss minimization***
S. Shalev-Shwartz and T. Zhang (2013)

Instead of solving the primal problem with (stochastic) gradient descent, solve the dual problem – with (stochastic) coordinate ascent (SDCA).

Submodular risk allocation
S. Ghamami and P. Glasserman (2017)

The problem of choosing which counterparty to use is of the form

$$\text{Minimize}_{A \subset S} F(A) + G(S \setminus A)$$

where S is the set of trades to execute, F and G are the collateral costs for counterparties 1 and 2, and A and $S \setminus A$ the trades sent to them. The costs could be of the form $\sigma(A) = \text{Sd}(\sum_{i \in A} X_i)$, which is submodular if $\text{Var } X$ is close to diagonal.

***A hitchhiker's guide
to automatic differentiation***
P.H.W. Hoffmann (2016)

Non-reversible Metropolis-Hastings J. Bierkens

A Markov chain on a discrete space S , with transition probabilities P , and a probability distribution π on S , satisfy the *detailed balance equation* if $\Gamma = 0$, where $\Gamma(x, y) = \pi(x)P(x, y) - \pi(y)P(y, x)$; the Markov chain is then said to be *reversible* and π is an invariant distribution. But the condition for π to be invariant is more general: $\Gamma \mathbf{1} = 0$ (*global balance*).

A *vorticity matrix* is a skew-symmetric matrix Γ such that $\Gamma \mathbf{1} = 0$. Given a reversible chain k with invariant distribution π and a vorticity matrix Γ ,

$$P(x, y) = k(x, y) + \frac{\Gamma(x, y)}{2\pi(x)}$$

defines a (non-reversible) chain with invariant distribution π (provided $P \geq 0$: since Γ has negative entries, this is not guaranteed).

The *non-reversible Metropolis-Hastings* (NRMH) chain

$$\begin{aligned} R(x, y) &= \frac{\Gamma(x, y) + \pi(y)Q(y, x)}{\pi(x)Q(x, y)} \\ A(x, y) &= \text{Max}\{1, R(x, y)\} \\ P(x, y) &= Q(x, y)A(x, y) \text{ if } x \neq y \end{aligned}$$

provided $Q(x, y) = 0 \Leftrightarrow Q(y, x) = 0$ and $\Gamma(x, y) \geq -\pi(y)Q(y, x)$, can be generalized to continuous spaces.

Non-reversible chains have better theoretical properties (lower variance, fewer large deviations).

The Langevin diffusion $dX = \nabla \log \pi dt + \sqrt{2}dW$ (for instance, the OU diffusion, $dX = -V^{-1}Xdt + \sqrt{2}dW$, which samples from $N(0, V)$) can be made non-reversible

$$dX = (I + S)\nabla \log \pi + \sqrt{2}dW,$$

where S is skew-symmetric; the MH acceptance step to correct the discretization error can be replaced by a NRMH step.

DeepWalk: online learning of social representations B. Perozzi et al. (2014)

Embed the vertices of a graph in \mathbf{R}^n (e.g., for automated text layout, or community discovery) as you would words (with skipgrams), considering random walks (of vertices) as “sentences” (or words).

Modeling dependence with C- and D-vine copulas: the R package CDVine E.C. Brechmann and U. Schepsmeier (2013)

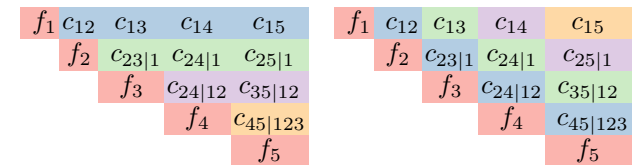
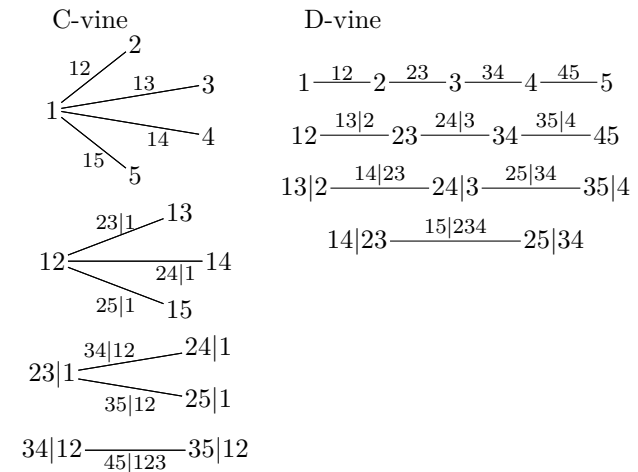
A multivariate distribution $f(x_1, x_2, x_3)$ can be decom-

posed as

$$\begin{aligned} f(x_1, x_2, x_3) &= f(x_1)f(x_2|x_1)f(x_3|x_1, x_2) \\ f(x_2|x_1) &= \frac{f(x_1, x_2)}{f(x_1)} \\ &= \frac{c_{12}(F_1(x_1), F_2(x_2))f(x_1)f(x_2)}{f(x_1)} \\ &= f(x_2)c_{12}(F_1(x_1), f_2(x_2)) \\ f(x_3|x_1, x_2) &= \frac{f(x_2, x_3|x_1)}{f(x_2|x_1)} \\ &= \frac{c_{23|1}(F(x_2|x_1), F(x_3|x_1))f(x_2|x_1)f(x_3|x_1)}{f(x_2|x_1)} \\ &= c_{23|1}(F(x_2|x_1), F(x_3|x_1))f(x_3|x_1) \\ &= c_{23|1}(F(x_2|x_1), F(x_3|x_1))f(x_3)c_{13}(F(x_1), F(x_3)) \\ f_{123} &= f_1f_2f_3c_{12}c_{13}c_{23|1} \end{aligned}$$

where $F(x_2|x_1)$ and $F(x_3|x_1)$ are known from c_{12} and c_{13} , and we assume that the copula density $c_{23|1}$ does not depend on x_1 .

There are many such decompositions. The $d(d-1)/2$ factors in those formulas can be arranged into $d-1$ trees – a **vine**. Special cases include stars (C-vines) and chains (D-vines), both defined by an ordering of the variables.



The **CDVine** package provides functions for BB copulas (which allow both upper and lower tail dependence, contrary to most archimedean copulas), bivariate copula analysis (**BiCop*** for contourplot, Kendall's plot, χ -plot, λ function, independence tests, gof tests, estimators (MLE, method of moments using Kendall's τ)), and C- and D-vine estimators (but you need to provide the variable order and the copula family).

Maximum likelihood estimation of mixed C-vines with applications to exchange rates
C. Czado et al. (2012)

To select the variable ordering in a C-vine copula, start with the most correlated variable,

$$i^* = \underset{i}{\operatorname{Argmax}} \sum_j |\tau_{ij}|,$$

renumber them so that $i^* = 1$, estimate the corresponding copulas C_{ij} , transform the variables $v_{j|1} = F(u_1|u_j)$, where $F(u|v) = \partial C_{uv}/\partial v$ and iterate.

Social media mining: an introduction
R. Zafarani et al. (2014)

2. The *Steiner tree problem*, finding the smallest tree, in a graph, containing a given subset of nodes, is NP-hard.

Signed graphs can model friend-foe relations; signed directed graphs are used in social status theory.

3. There are many node *centrality* measures:

- Eigenvector: $c_i = \lambda^{-1} \sum a_{ij}c_j$, *i.e.*, c is an eigenvector for eigenvalue λ (from the Perron-Frobenius theorem, the largest eigenvalue gives positive centralities);
- Katz and PageRank are variants of eigenvector centrality;
- Betweenness: number of shortest paths a node is on;
- Closeness: (harmonic) average distance to the other nodes.

There are a few more node metrics:

- Clustering: proportion of triangles (for signed graphs, one can distinguish between balanced and unbalanced triangles);
- Reciprocity (for directed graphs).

There are many node similarity measures:

- Number of nodes in common, $|N_x \cap N_y|$;
- Jaccard similarity, $\frac{|N_x \cap N_y|}{|N_x \cup N_y|}$;
- Cosine similarity, $\frac{|N_x \cap N_y|}{\sqrt{|N_x||N_y|}}$;
- Regular similarity, $\sigma_{ij} = \alpha \sum_{k\ell} A_{ik}A_{i\ell}\sigma_{k\ell}$;
- Adamic-Adar, $\sum_{z \in N_x \cap N_y} \frac{1}{\log |N_z|}$;
- Preferential attachment, $|N_x||N_y|$;
- Katz (weighted average path length);
- Hitting time (average time a random walk takes to reach y from x), commute time;
- Rooted PageRank;
- SimRank, $\sigma_{xy} = \lambda \frac{1}{|N_x||N_y|} \sum_{x' \in N_x} \sum_{y' \in N_y} \sigma_{x'y'}$.

4. Network models try to reproduce the following stylized facts:

- Power law degree distribution (*scale-free* network);
- High clustering coefficient;

- Low average path length (*small world*).

Random graphs models, $G(n, p)$ (n nodes, edge probability p) and $G(n, k)$ (n nodes, k edges) show a phase transition (apparition of a giant component) when the average degree reaches 1. The degree distribution is binomial (asymptotically Poisson – not a power law) and the clustering coefficient, p , is low. The average path length is $\log \# \text{nodes} / \log \langle \text{degree} \rangle$.

The *small world* model starts with a regular (ring) lattice and rewrites each edge (replaces one extremity at random) with probability β . The degree distribution is still incorrect, but the clustering coefficient and the average path length are fine.

The *preferential attachment* (Barabási-Albert) model adds one node at a time, connects it to m nodes, chosen with probabilities proportional to their degrees. The resulting graph is scale-free, *i.e.*, its degree distribution is a power law, but its exponent is always the same (3); the average path length is small enough, but the clustering coefficient is too low.

The *stochastic block model* was not mentioned.

5. The *weighted vote relational neighbour* (wvRN) classifier is similar to k -NN:

$$P[y_i = 1] = \frac{1}{|N_i|} \sum_{j \in N_i} P[y_j = 1]$$

(since not all y_i 's are observed, iterate to find a fixed point).

To evaluate the result of a clustering algorithm:

- Cohesiveness: $\sum_{i \in \text{Clusters}} \sum_{j \in C_i} \|x_j - c_i\|^2$
- Separateness: $\sum_i \|c_i - c\|^2$
- Silhouette: $\frac{1}{n} \sum_x \frac{b(x) - a(x)}{\max\{b(x), a(x)\}}$, where $a(x)$ is the average distance to the other elements of the cluster and $b(x)$ the minimum average distance to the elements of another cluster.

6. Communities could be defined as *cliques*, (but they are rare in real networks), k -plexes (subgraph with n nodes, each of degree at least $n - k$) or k -cliques (subgraphs in which all shortest paths have length at most k).

The clique percolation method (CPM) uses cliques as seeds: communities are the connected components of the clique graph, whose vertices are cliques of size k and with edges when cliques share $k - 1$ nodes.

Classical clustering algorithms can be used with node similarity.

Minimum cuts tend to generate singletons. Instead, one can look for partitions $P = (P_1, \dots, P_k)$ minimiz-

ing

$$\text{RatioCut}(P) = \frac{1}{k} \sum \frac{\text{Cut}(P_i, \bar{P}_i)}{|P_i|}$$

$$\text{or NormalizedCut}(P) = \frac{1}{k} \sum \frac{\text{Cut}(P_i, \bar{P}_i)}{\text{vol}(P_i)}$$

where the volume is the sum of the degrees. Those problems are NP-hard, but can be relaxed (spectral clustering).

The *modularity* of a community is the difference between the number of edges inside it and the expected number of edges for a random graph with similar degrees,

$$\text{Modularity}(P) = \sum_i \sum_{j,k \in P_i} A_{jk} - \frac{d_j d_k}{2m}.$$

The *Girvan-Newman* divisive hierarchical communities are obtained by progressively removing edges with the largest edge betweenness.

Evolving networks have a few more stylized facts:

- Segmentation into giant component, stars and singletons;
- Densification: the number of edges increases faster than the number of nodes, $|E_t| \propto |V_t|^\alpha$, $\alpha > 1$ (often, $\alpha \approx 1.6$);
- Diameter shrinkage.

To find evolving communities, consider a single optimization problem, for the communities of G_t for all t 's, with a penalty to keep the communities of G_t and G_{t+1} close.

7. The information cascade model (ICM) models information diffusion on a known network, in a sender-centric fashion: at each time step, each activated node (potential client who has already heard of our product) activates nearby nodes (talks to them and succeeds if his conviction p_{ij} exceeds their threshold t_j). Choosing the seed nodes to maximize the speed of cascades is a submodular optimization problem.

The diffusion of innovation (on a complete or unknown network) can be modeled as

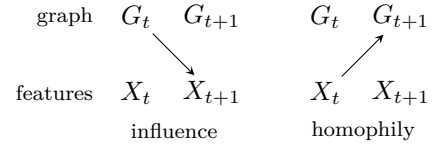
$$A' = (\alpha + \beta A)(\text{Population} - A)$$

(where A is the number of adopters, α the external influence and β the internal influence) or with epidemic models (SIRS – SI, SIR, SIS are obtained with $\lambda = \gamma = 0$, $\lambda = 0$ and $\lambda = \gamma$, $R = 0$)

$$\begin{aligned} S' &= \lambda R - \beta IS \\ I' &= \beta IS - \gamma I \\ R' &= \gamma I - \lambda R \\ N &= S + I + R \end{aligned}$$

8. *Assortativity* measures the propensity of similar nodes to be linked. For categorical features, it can be assessed by modularity; for continuous ones, by covariance or correlation.

It can be explained by *influence* (the presence of links changes node attributes) or *homophily* (feature similarity creates edges).



Influence can be measured by centrality, or the size of the cascade (in the ICM or SIR model), e.g., the number of nodes affected, the average number of hops, or the rate at which the population is infected. Specific examples include the number of in-links, the inverse of the number of outlinks (novelty), the number of comments, the length for blogposts, or the number of followers (popular but imprecise), retweets and mentions for Twitter.

Contrary to sender-centric cascade (epidemic) models, influence (or threshold) models are receiver-centric.

In the *linear threshold model* (LTM, e.g., Schelling's model of racial segregation), node i becomes active if $\sum_{j \in N_i} w_{ji} \geq \theta_i$.

The *linear influence model* (LIM)

$$|P_t| = \sum_{u \in P_t} I(u, t - t_u) \quad \text{population influenced}$$

$$I(u, t - t_u) = c_u(t - t_u)^{-\alpha_u} \quad \text{influence function}$$

$$|P_t| = \sum_{u,t} A_{ut} I_{ut}, \quad I \geq 0$$

can be solved with nonnegative least squares.

Several tests disentangle influence and homophily:

- Shuffle test: compare the **social correlation** (the slope in the logistic regression activation = $\alpha + \beta \times \# \text{ active friends} + \varepsilon$) before and after shuffling the activation times;
- Edge reversal test: the social correlation should not change when reversing the edges;
- Randomization test: look at the assortativity gain

$$\begin{aligned} \text{Gain}_{\text{influence}}(t) &= A(G_t, X_{t+1}) - A(G_t, X_t) \\ \text{Gain}_{\text{homophily}}(t) &= A(G_{t+1}, X_t) - A(G_t, X_t). \end{aligned}$$

9. Recommendation engines can average the scores for similar items and/or users, but most rely on matrix factorizations, finding embeddings U (for users) and I (items) such that $\text{similarity}(i, j) \approx \cos(U_i, I_j)$, sometimes with a penalty $\text{sim}(i, j) \|U_i - U_j\|^2$ to ensure nearby nodes have nearby feature vectors.

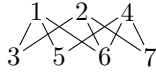
10. *Link prediction* can be based on node similarity. User migration can be forecasted from user activity, network size and rank (in-degree).

Error correcting codes F. Lemmermeyer (2005)

Given a finite set A (alphabet, e.g., $\{0, 1\}$), a *code* is a subset $C \subset A^n$.

A *linear code* is a linear subspace of \mathbf{F}_q^n .

The Hamming [7, 4] code, in \mathbf{F}_2^7 , adds 3 parity bits to a 4-bit message.



A code of type $[n, k, d]_q$ is a subset of \mathbf{F}_q^n of size q^k , with minimum distance d .

The *minimum distance* of a code is the minimum Hamming distance between any two codewords. The minimum distance of a linear code is the minimum number of non-zero bits in a codeword (the minimum weight). A code with minimum distance d can detect $d-1$ errors and correct $\lfloor (d-1)/2 \rfloor$.

[The linear algebra notation for linear codes often uses row vectors.] A *generator matrix* of a linear code C is a matrix G whose rows form a basis of C ; it can be used to encode messages,

$$\begin{cases} \mathbf{F}_q^k & \longrightarrow \mathbf{F}_q^n \\ x & \longmapsto xG. \end{cases}$$

A *parity matrix* of C is a generator matrix H of C^\perp ; it can be used to detect the presence of errors:

$$C = \{c \in \mathbf{F}_q^n : cH' = 0\}.$$

Codewords are elements c whose *syndrome* cH' is zero.

The Hamming code $\text{Ham}(r)$ is defined by a parity matrix whose columns are the nonzero vectors in \mathbf{F}_2^r ; it has type $[2^r - 1, 2^r - 1 - r, 3]_2$.

For a code $C \subset \mathbf{F}_q^n$ with minimum distance $2t + 1$, the *sphere packing bound* is

$$|C| \sum_{i=0}^t \binom{n}{i} (q-1)^i \leq q^n.$$

For a *perfect code*, it is an equality. The perfect codes are Hamming ($t = 1$), repetition codes of odd length $n = 2t + 1$ and the binary Golay code G_{23} .

For linear codes of type $[n, k, d]$, the *singleton bound* is $d \leq n - k + 1$.

If the generator matrix is in standard form, $G = (I|A)$, a corresponding parity check matrix is $H = (-A'|I)$.

To decode a message c encoded with a general linear code $C \subset \mathbf{F}_q^n$, notice that two words x, y are in the same coset $v + C$ iff they have the same syndrome. The *coset leader* e of a coset $v + C$ is the word in it with minimal weight: subtracting it from the encoded message corrects the errors (for efficient decoding, one can store a mapping table from syndromes to coset leaders).

The *Plotkin* product of two codes $C_1, C_2 \subset \mathbf{F}_2^n$ of the same length is

$$C_1 * C_2 = \{(u, u + v) : u \in C_1, v \in C_2\}.$$

Cyclic linear codes are linear codes stable by cyclic permutations

$$(c_1 c_1 \cdots c_{n-1}) \longmapsto (c_{n-1} c_0 c_1 \cdots c_{n-2});$$

they correspond to ideals of $\mathbf{F}_q[X]/(X^n - 1)$. Those ideals are principal and generated by the divisors g of $X^n - 1$. If $X^n - 1 = gh$, then h is a parity check polynomial. To decode a message, compute its euclidean division by h : the syndrome polynomial $s(X)$ is the remainder. The error is the word of minimum weight among the cyclic shifts $X^i s(X) \bmod g(X)$.

If r consecutive powers of an n th root of unity are roots of $g(X)$, then $d \geq r + 1$.

Golay's G_{23} code, of type $[23, 12, 7]_2$, corresponds to

$$X^{11} + X^9 + X^7 + X^6 + X^5 + X + 1 \mid X^{23} - 1.$$

A tutorial on conformal prediction G. Shafer and V. Vovk (2013)

Conformal prediction regions are defined by $y \in R_\varepsilon$ if

$$p_y = \frac{\#\{I : \alpha_i > \alpha_n\}}{n} > \varepsilon$$

where $\alpha_i = d(\{z_1, \dots, z_n\} \setminus \{z_i\}, z_i)$, $z_k = (x_k, y_k)$ for $k < n$, $z_n = (x_n, y)$, d is some *non-conformity measure* and $\{\dots\}$ denotes multisets.

They are valid, *i.e.*, they contain the true value the advertized proportion of the time, provided the sequence of random variables is *exchangeable*.

The bags of observations can be seen as an *online compression model*, with summarizing and updating operators,

$$\begin{aligned} \{z_1, \dots, z_n\} &\longmapsto \{z_1, \dots, z_n\} \\ \{z_1, \dots, z_n\}, z_{n+1} &\longmapsto \{z_1, \dots, z_n, z_{n+1}\}. \end{aligned}$$

Relaxations of exchangeability include within-label exchangeability or the online Gaussian linear model.

Advances in financial machine learning M. Lopez de Prado (2018)

2. Financial data is often irregularly-spaced and should be transformed into tabular format, often as *bars*, *i.e.*, summaries (open, high, low, close, VWAP) over intervals: time bars, tick bars (every 1000 ticks), volume bars (number of shares), dollar bars (value traded), tick/volume/dollar imbalance bars (cumulated order imbalance),

$$b_t = \begin{cases} b_{t-1} & \text{if } \Delta \text{price} = 0 \\ \text{sign}(\Delta \text{price}) & \end{cases}$$

tick/volume/dollar runs bars (length of the longest run in b_t).

[The book tends to use ratio returns instead of log-returns (ratio returns are skewed and have extreme positive values) and cumulated products instead of cumulated sums of logarithms (this is numerically problematic).]

3. Instead of forecasting the returns, or their sign, or whether they will exceed some threshold, h bars ahead (with dollar bars, otherwise h should vary), use a *triple barrier*: forecast whether the price will go above or below (stop-loss) a threshold before time h – there are three outcomes: profit-taking, stop-loss and time-loss.

Consider using a *cascade* of models: a first model with high recall, followed by another with high precision (e.g., separately predicting the sign of the returns and whether the time barrier will be hit) [he calls that “met-labeling”].

4. Using bars leads to overlaps: weigh the observations accordingly.

5. Returns may not be the only way of turning log-prices into stationary time series, as required by many statistical procedures: *fractional integration*, which accounts for long memory, may be a better choice.

6,7. Use ensemble methods, cross-validation (with no overlap, and perhaps even an embargo).

8. Feature importance can be assessed by mean decrease impurity (in-sample) or mean decrease accuracy (out-of-sample). These are single-feature importance measures: to limit the substitution effect, try to orthogonalize the features.

9. [The discussion of hyperparameter tuning only mentions grid and random search, not low-discrepancy sequences or Bayesian optimization.]

10. Size your position to account for the possibility that the signal will strengthen further using, e.g., another model to estimate the probability of a correct forecast.

11. Backtesting can be dangerous: compute the *probability of backtest overfitting*.

12. While k -fold cross-validation suffers from look-ahead bias, it should not be completely discarded: a model able to predict the future from the past but not the past from the future would be suspicious.

Account for multiple testing with FWER or FDR.

13. Backtesting on synthetic data, and comparing the strategy with a random one, can help check if the model overfits.

14. Backtest statistics should include time (different regimes should be present), capacity, leverage, frequency of bets, average holding period, annual turnover, correlation to the underlying (to check if the strategy is just long or short); usual performance measures, P&L from hits, from misses; Herfindahl index of the time between bets, size of the tails (e.g., $\langle r_+ \rangle$,

$\langle r_- \rangle$); Sharpe ratio variants (correction for short time series, skewness, kurtosis, multiple testing); attribution.

15. The *strategy risk* is the sensitivity of the Sharpe ratio of a strategy (say, paying π with probability p and $-\pi$ with probability $1 - p$, with n bets per year) to the hit ratio p , or by how much the hit ratio should drop to wipe out the profit.

17. Structural breaks can be detected with CUSUM tests, looking if the cumulated normalized residuals of $y \sim x$ are consistent with a constant slope, or Dickey-Fuller-like tests.

- To test for explosive behaviour in $y_t = \rho y_{t-1} + \varepsilon_t$, i.e., $\rho = 1$ if $t \leq t_0$ and $\rho > 1$ if $t > t_0$, test for a non-zero slope in $\Delta y_t \sim y_{t-1} \mathbf{1}_{t > t_0}$;
- The ADF test uses more lags $\Delta y_t \sim y_{t-1} \mathbf{1}_{t > t_0} + \sum_{\ell} \Delta y_{t-\ell}$;
- The supremum ADF test uses $\sup_{t_0} \text{ADF}_{t_0, t}$, but there are quantile and “conditional” (cf. CVaR) variants.

Sub- and super-martingale tests test $H_0 : \beta = 0$ versus $H_1 : \beta \neq 0$ on

$$y = \alpha + \gamma t + \beta t^2 + \varepsilon$$

$$\log y = \alpha + \gamma t + \beta t^2 + \varepsilon$$

$$\log y = \alpha + \beta t + \varepsilon$$

$$\log y = \alpha + \beta \log t + \varepsilon$$

and use $\sup_{t_0} |\hat{\beta}/\sigma_{\beta}|$.

19. There have been three generations of microstructure features: first-generation models assume the mid-price is a random walk, or use high and low values to estimate volatility, spread or imbalance; second-generation models focus on illiquidity,

Kyle	$\Delta \text{price} \sim \text{signed volume}$
Amihud	$ \Delta \log \text{price} \sim \text{volume}$
Hasbrouk	$\Delta \log \text{price} \sim \pm \sqrt{\text{volume}}$;

third-generation models (PIN, VPIN) focus on trades.

Other microstructural features include the distribution of order sizes, cancellation rates, limit orders, market orders, presence of TWAP traders, options markets (option quotes contain little information, but option trades are informative), serial correlation of the signed order flow.

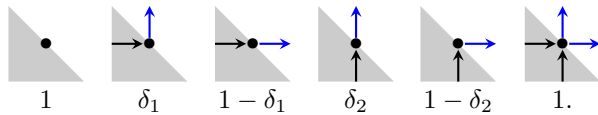
Microstructural information can be defined by building a model to forecast a market maker’s profit from those features and looking at its performance (e.g., cdf of the log-likelihood).

22. To process big data, scientists tend to use HPC clusters rather than the cloud: the cloud has distributed storage, relies on virtualization (flexible, but with a performance impact), and is less scalable (because of IO and network virtualization) and more expensive than HPC.

Convergence
of the stochastic six-vertex model to the ASEP
A. Aggarwal

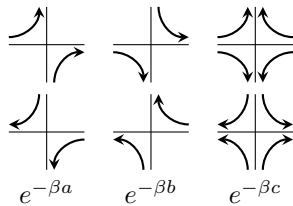
In the asymmetric simple exclusion process (ASEP) model, particles are initially placed on the integers (not all of them) and move left or right, if the new position is not occupied, at times given by random exponential clocks with rates $0 \leq L < R$.

The *six-vertex model* is a probability distribution on the set of paths in the first quadrant $\mathbf{Z}_{\geq 0}^2$ starting from a set of points on the x or y axis, obtained by extending the paths from $T_n = [x + y \leq n]$ to T_{n+1} with probabilities



A numerical study of the F-model
with domain-wall boundaries
R. Keesman and J. Lamers

The six-vertex model on a square lattice demands exactly two ingoing and two outgoing edges at each vertex.



Modified profile likelihood inference
and interval forecast
of the burst of financial bubbles
V. Filimonov et al. (2016)

The LPPL model can be justified by a geometric random walk with jumps

$$\frac{dp}{p} = \mu_t dt + \sigma_t dW - \kappa dJ$$

where the hazard rate of the jumps J grows exponentially with log-periodic oscillations

$$h(t) = \alpha(t_c - t)^{m-1} [1 + \beta \cos(\omega \log(t_c - t) - \phi')].$$

If the excess return μ_t is proportional to the hazard rate, then the expected log-price is

$$E[\log p] = A + B(t_c - t)^m + C(t - t_c)^m \cos(\omega \log(t_c - t) - \phi).$$

We expect

$$0.1 \leq m \leq 0.9, \quad 6 \leq \omega \leq 13, \quad B < 0, \quad D = \frac{m|B|}{\omega|C|} \geq 0.8.$$

In the likelihood L , one can separate the critical time t_c from the nuisance parameters η ,

$$\hat{t} = \underset{t}{\operatorname{Argmax}} \underset{\eta}{\operatorname{Max}} L(t, \eta)$$

where $L_p(t) = \operatorname{Max}_{\eta}(t, \eta)$ is the **profile likelihood**. The likelihood can sometimes be factored into *marginal* and *conditional* likelihoods, $L(t, \eta) = L(t)L(\eta)$, but not here.

The profile likelihood ignores the uncertainty on the nuisance parameters. The *modified profile likelihood* adds a penalty using the Fisher information matrix of the nuisance parameters.

t : parameter of interest

η : nuisance parameters

$L(t, \eta)$: likelihood

$\ell(t, \eta) = \log L(t, \eta)$ log-likelihood

$\ell_p(t) = \operatorname{Max}_{\eta} \ell(t, \eta)$ profile log-likelihood

$\hat{\eta}_t = \operatorname{Argmax}_{\eta} \ell(t, \eta)$

$\hat{t}, \hat{\eta} = \operatorname{Argmax}_{t, \eta} \ell(t, \eta)$

$I(\hat{\eta}_t) = - \frac{\partial^2 \ell}{\partial \eta \partial \eta'} \Big|_{\eta = \hat{\eta}_t}$ Fisher information

$|A| = |\det A|$

$\ell_m(t) = -\frac{1}{2} \log |I(\hat{\eta}_t)| + \log J(t) + \ell_p(t)$

$J(t)$: Jacobian term, to make the modified profile likelihood invariant to a reparametrization of the nuisance parameters (difficult to evaluate)

$J(t) \approx \frac{|I(\hat{\eta}_t)|}{|\Sigma(t, \hat{\eta}_t; \hat{t}, \hat{\eta})|}$

$\Sigma(t_1, \eta_1; t_2, \eta_2) = E_2 \left[\frac{\partial \ell}{\partial \eta} \Big|_{t_1, \eta_1} \frac{\partial \ell}{\partial \eta'} \Big|_{t_2, \eta_2} \right]$

$\frac{\partial \ell}{\partial \eta}$: score function

E_2 : expectation wrt the distribution of the error term $\varepsilon(t_2, \eta_2)$

The *multiscale modified profile likelihood* plots those confidence intervals in the $t_2 - t_1$ vs $t_c - t_2$ space, highlighting when the constraints are satisfied.

Dissection of Bitcoin's
multiscale bubble history
J.C. Gerlach et al. (2018)

To identify Bitcoin bubbles ex post:

- Look for “drawups”, successions of positive returns interrupted by negative returns no larger than ε , immediately followed by a similarly-defined drawdown, with $\varepsilon = \varepsilon_0 \sigma$, where σ is the volatility over the past w days, and keep dates identified as a peak by 95% of the parameters $(\varepsilon, w) \in [0.1, 5] \times [10, 60]$ (“epsilon drawdown method”);
- Define the begining of the bubbles by fitting an LPPL model with various starting points, and keeping that with the lowest root mean square error, detrended by removing the effect of the estimation period $t_2 - t_1$ (“Lagrange regularization”);

- Define the end of the post-bubble correction as the minimum before the next bubble.

For real-time bubble prediction, fit the LPPL model on different window sizes and cluster the $(t_2 - t_1, t_c - t_1)$, with k -means and the silhouette to choose the number of clusters, to have one or several scenarios.

***Blockchain: data malls,
coin economies and keyless payments***
Z. Kakushadze and R.P. Russo (2018)

The Iota data marketplace is a decentralized data lake whose coin allows users to buy and sell data from each other. Augur is a decentralized prediction market based on Ethereum, using a “reputation” token. Golem is a network to rent spare computing power. MicroRaiden provides off-blockchain micropayments on Ethereum. The Everipedia (a Wikipedia competitor) token is mined by editors by making valuable contributions.

Data provenance, keyless payments (sending BTC to an email address, in a decentralized way) and voting are other applications.

***Cryptocurrency portfolio management
with deep reinforcement learning***
Z. Jiang and J. Liang

End-to-end portfolio construction to find the weights maximizing the after-cost returns, on 12 (liquid) cryptocurrencies, from the prices on 50 30-minute periods, with a convolutional network (CONV+FC). [There is no “reinforcement learning”; the convolution kernels are 12×4 , *i.e.*, they also convolve in the “asset” direction.]

***Are Bitcoin bubbles predictable?
Combining a generalized Metcalfe’s law
and the LPPLS model***
S. Wheatley et al. (2018)

Metcalfe’s law states that the value of a network is proportional to the square of its number of nodes. The Bitcoin market value is proportional to some power (1.7) of the number of users (or the number of transactions, from bitcoininfocharts.com, smoothed): deviations from this generalized Metcalfe’s law is a bubble signal; it can be combined with the LPPL model.

***Classification of crypto-coins and tokens
from the dynamics
of their power law capitalization distributions***
K. Wu et al. (2018)

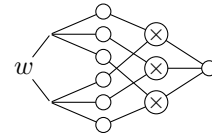
The distribution of the market capitalization of coins (resp. tokens), from coinmarketcap.com, is well-described by a Pareto (power law) distribution with exponent 0.6 (resp. 1.1). This can be explained by modeling the market capitalization of each coin with independent geometric Brownian motions (with the same parameters), with new coins created by a Poisson process with exponentially growing intensity, and existing coins dying with a constant hazard rate.

***Pricing options
with exponential Levy neural network***
J. Huh (2018)

For Fourier-transform-based option pricing, model the Lévy density

$$\frac{d\nu}{dx}(x) = e^{-x} \mathcal{F}^{-1}[h(w)](x)$$

with a 1-hidden-layer neural net h whose hidden nodes are in two groups, whose outputs are multiplied.



***Generating virtual scenarios
of multivariate financial data
for quantitative trading applications***
J. Franco-Pedroso et al. (2018)

Simulated returns from traditional models are not realistic (Gaussian returns) and/or not scalable (multivariate GARCH, *e.g.*, BEKK). To reproduce “multivariate stylized facts” (kurtosis, skewness, volatility clustering, increased correlation in periods of high volatility, number of “trends”, distribution of the “trend ratios” μ/σ), take a more empirical approach:

- Split the data into up- and down-trends for the market index;
- In each of those trends, estimate the variance matrix on a moving window;
- Sample Gaussian returns from those distributions [I suspect they reject the samples disagreeing with the trend].

***Forecasting directional changes
in financial markets***
A. Bakhach et al (2015)

An upward or downward trend can be defined as a price change beyond a threshold θ ; trends go from one extreme to another; a confirmation point is the first point, after an extreme, exceeding the threshold. The overshoot is the difference between an extreme and the previous confirmation point (rescaled by θ). One can predict of the overshoot of the current trend will exceed some threshold d using decision trees and three variables: the Aroon up and down indicators, *i.e.*, the number of up (or down) trends ago the maximum (or minimum) extremum (or confirmation point) was (among the past $n = 20$ up/down trends) and the difference between the last two confirmation points.

***The role of the dynamic conditional
quartic beta and the capital markets***
G. Corvasce

Define a (cross-sectional, time-varying) *quartic beta* from the quartic loss $\sum (y_i - \beta x_i)^4$.

**Imaging time series
to improve classification and imputation
Z. Wang and T. Oates (2015)**

Convert a time series to an image, using a *Gramian angular field*:

- Rescale x to $[-1, 1]$;
- Let $\phi_t = \arccos x_t$;
- Consider $\cos(\phi_t + \phi_s)$ or $\sin(\phi_t - \phi_s)$

or a *Markov transition field*

- Bin the observations, $y = \text{discretize}(x)$;
- Compute the transition matrix P ;
- Spread it out to preserve the temporal dependence

$$M_{t \rightarrow s} = P_{y_t \rightarrow y_s}$$

and use a (tiled) CNN.

**Deep reinforcement learning bootcamp
P. Abbeel et al. (2017)**

1. Planning or optimal control is the search of the optimal policy on a *known* Markov decision problem (MDP).

Value iteration iterates the Bellman equation

$$\begin{aligned} V_0(s) &= 0 \\ V_{k+1}(s) &= \max_a \mathbb{E}_{s'} [R(s, a, s') + \gamma V_k(s')] \\ &= \max_a \sum_{s'} P(s'|a, s) [R(s, a, s') + \gamma V_k(s')] \end{aligned}$$

Q-value iteration is similar, but computes $Q(s, a)$, the expected value, if we start in state s , take action a , and act optimally thereafter.

$$Q_{k+1}(s, a) = \sum_{s'} P(s'|a, s) [R(s, a, s') + \gamma \max_{a'} Q(s', a')]$$

These are just systems of equations but, because of the maximum, they are not linear. If the policy π is fixed, however, it is just a linear system, which can be solved iteratively or directly (**policy evaluation**).

$$V^\pi(s) = \sum_{s'} P(s'|a, s) [R(s, a, s') + \gamma V^\pi(s')]$$

Policy iteration alternates two steps:

- Compute the value V^π of the policy π_k ;
- Improve the policy by taking the best 1-step action

$$\pi_{k+1}(s) = \underset{a}{\text{Argmax}} P(s'|a, s) [R(s, a, s') + \gamma V^\pi(s')]$$

2. If the MDP is not known, we can act at random and estimate the expected Q -value (**tabular Q-learning**).

$$\begin{aligned} Q(s, a) &\leftarrow (1 - \alpha)Q(s, a) + \alpha \cdot \text{target} \\ \text{target} &= R(s, a, s') + \gamma \max_{a'} Q(s', a') \end{aligned}$$

This is *off-policy* learning: the first term comes from the policy actually followed (e.g., ϵ -greedy), the second

from the optimal policy found so far. The learning rate α should satisfy $\sum \alpha_t = \infty$, $\sum \alpha_t^2 < \infty$.

Approximate Q-learning replaces the table $Q(s, a)$ with a function (e.g., a linear combination of hand-selected features), and a gradient update

$$\theta \leftarrow \theta - \frac{1}{2} \alpha \nabla_\theta (Q_\theta(s, a) - \text{target})^2.$$

3. Deep Q networks (DQN) use a neural network to model Q_θ , but

- The target is not stationary;
- The data is not iid;

To address those concerns:

- Use batches (*experience replay*);
- Use an older set of weights for the target (*target network*);
- Use Huber loss instead of square loss;
- Use RMSProp instead of standard SGD;
- Anneal the exploration rate.

Neural fitted Q-iteration is a batch version of DQN: generate a lot of ϵ -greedy episodes; fit the resulting target for a while; iterate.

Double DQN uses two sets of weights, to select the best action, and to estimate it (otherwise $\max_a Q_\theta(s, a)$ is biased upwards)

Prioritized experience replay uses the Bellman error as transition weights.

$$\left| r + \gamma \max_{a'} Q_{\theta_1}(s', a) - Q_{\theta_2}(s, a') \right|$$

In a *duelling DQN*, the neural network adds structure to the Q -value and separately forecasts state value and advantage, $Q(s, a) = V(s) + A(s, a)$.

Noisy nets add noise to the parameters for better exploration.

4. Policy gradient methods directly look for an optimal stochastic policy $\pi_\theta(s)$ (a deterministic policy is a combinatorial object, more difficult to optimize). This is on-policy (less exploration) and less sample-efficient, but can deal with large action spaces (with Q -learning, $\text{Argmax}_a Q(s, a)$ can be difficult to compute).

Consider a whole episode and let

$$\tau = (s_0, u_0, s_1, u_1, \dots, s_N, u_N)$$

be a state-action sequence. We want to maximize the

expected utility $U(\theta) = \sum P_\theta(\tau)R(\tau)$.

$$\begin{aligned}\nabla_\theta U &= \sum \nabla_\theta P_\theta(\tau)R(\tau) \\ &= \sum P_\theta(\tau) \frac{\nabla_\theta P_\theta(\tau)}{P_\theta(\tau)} R(\tau) \\ &= \sum P_\theta(\tau) \nabla_\theta \log P_\theta(\tau) R(\tau) \\ &= \mathbb{E}[\nabla_\theta \log P_\theta(\tau) R(\tau)] \\ &\approx \frac{1}{m} \sum_i \nabla_\theta \log P_\theta(\tau_i) R(\tau_i)\end{aligned}$$

$$\begin{aligned}\nabla_\theta \log P_\theta(\tau) &= \nabla_\theta \log \prod_t P(s_{t+1}|s_t, a_t) \pi_\theta(s_t|s_t) \\ &= \sum_t \nabla_\theta \log P(s_{t+1}|s_t, a_t) + \nabla_\theta \log \pi_\theta(a_t, s_t) \\ &= \sum_t \nabla_\theta \log \pi_\theta(a_t|s_t)\end{aligned}$$

This can also be derived using *importance sampling*:

$$\begin{aligned}U(\theta) &= \mathbb{E}_{\tau \sim \theta} [R(\tau)] \\ &= \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{p_\theta(\tau)}{p_{\theta_{\text{old}}}(\tau)} R(\tau) \right] \\ \nabla U(\theta)|_{\theta=\theta_{\text{old}}} &= \mathbb{E}_{\tau \sim \theta_{\text{old}}} \left[\frac{\nabla_\theta p_\theta(\tau)|_{\theta=\theta_{\text{old}}}}{p_{\theta_{\text{old}}}(\tau)} R(\tau) \right] \\ &= \mathbb{E}_{\tau \sim \theta_{\text{old}}} [\nabla_\theta \log P_\theta(\tau)|_{\theta=\theta_{\text{old}}} R(\tau)]\end{aligned}$$

To reduce variance, one can add a baseline

$$\begin{aligned}\nabla U(\theta) &\approx \frac{1}{m} \sum_i \nabla_\theta \log P_\theta(\tau_i) R(\tau_i) \\ &\approx \frac{1}{m} \sum_i \nabla_\theta \log P_\theta(\tau_i) (R(\tau_i) - b)\end{aligned}$$

and discard terms that do not depend on the current action

$$\begin{aligned}\nabla U(\theta) &\approx \frac{1}{m} \sum_i \sum_t \nabla_\theta \log \pi_\theta(a_{it}, s_{it}) (R(\tau_i) - b) \\ &= \frac{1}{m} \sum_i \sum_t \nabla_\theta \log \pi_\theta(a_{it}, s_{it}) \left(\sum_{k \geq t} R(s_{ik}, a_{ik}) - b \right).\end{aligned}$$

The difference $A_{it} = \sum_{k \geq t} R(s_{ik}, a_{ik}) - b$ is the **advantage**.

Here are possible choices for the baseline:

- Constant: $b = \mathbb{E}[R(\tau)] \approx \frac{1}{m} \sum R(\tau_i)$;
- Optimal constant,

$$b = \frac{\sum (\nabla \log p(\tau_i))^2 R(\tau_i)}{\sum (\nabla \log p(\tau_i))^2};$$

- Time-dependent:

$$b = \mathbb{E}[R(\tau_{\geq t})] \approx \frac{1}{m} \sum_i \sum_{k \geq t} R(s_{ik}, a_{ik});$$

- State-dependent (**actor-critic** – in particular, if the current state is hopeless, regardless of the actions taken, there is no valuable information and the gradient is zero):

$$b(s_t) = \mathbb{E}[r_t + r_{t+1} + \dots] = V^\pi(s_t).$$

To estimate the value function $V^\pi(s)$, we can use supervised learning, e.g.,

- Monte-Carlo: minimize

$$\left[V_\phi^\pi(s_{it}) - \sum_{k \geq t} R(s_{ik}, u_{ik}) \right]^2$$

(we end up with two neural nets: one for the policy, one for the value function);

- Temporal difference (TD): use the Bellman equation for V^π

$$\begin{aligned}\phi_{i+1} &= \underset{\phi}{\text{Argmin}} \sum \left\| R + \gamma V_{\phi_i}^\pi(s_{t+1}) - V_\phi^\pi(s_t) \right\|_2^2 + \\ &\quad \lambda \|\phi - \phi_i\|^2\end{aligned}$$

(there are now two hyperparameters, γ and λ).

To reduce the variance:

- Discount more;
- Use function approximation:

$$\begin{aligned}Q^\pi(s, u) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \dots] &= Q_0 \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1)] &= Q_1 \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2)] &= Q_2 \\ &= \mathbb{E}[r_0 + \gamma r_1 + \dots + \gamma^4 r_4 + \gamma^5 V^\pi(s_5)].\end{aligned}$$

The **Async advantage actor critic** (A3C) uses Q_5 ; the *generalized advantage estimation* (GAE) averages all of those:

$$Q^\pi = (1 - \lambda) \sum_{n \geq 0} \lambda^n Q_n^\pi.$$

5. Notice that the policy gradient step is also the gradient step of some optimization problem: we can replace that step with a few iterations of an optimization algorithm solving that problem, with a constraint or penalty to limit the step size.

More precisely, the gradient $E_t[\nabla_\theta \log \pi(a|s)A]$ could come from the surrogate loss $E_t[\log \pi(a|s)A]$ or to the importance sampling loss

$$E_t \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A \right].$$

Trust region policy optimization (TRPO) uses a constraint to define the “trust region” (the region where we trust our approximation to be close enough)

$$E[\text{KL}(\pi_{\theta_{\text{old}}}(\cdot|s), \pi_\theta(\cdot|s))] \leq \delta.$$

In practice, use a linear approximation of the objective, a quadratic approximation of the constraint (the Kullback-Leibert divergence becomes the Fisher information matrix – it is a **natural gradient**) and solve

using the conjugate gradient (Hessian-free optimization).

Proximal policy optimization (PPO) uses a penalty instead of a constraint – but adjusting the level of the penalty is harder than adjusting the size of the trust region – a constant value does not work.

Those methods do not work well with CNNs, RNNs and complicated architectures.

Variants include:

- KFAC: Natural policy gradient blockwise approximation of the Fisher information matrix;
- ACKTR: A2C + KFAC natural gradient;
- PPO with clipped objective.

6. Practical advice:

- Use several baselines: cross-entropy, policy gradient, Q-learning SARSA;
- Use more samples per batch;
- Check the sensitivity to all parameters – the method should be robust;
- Try different random seeds;
- Standardize: $\text{clip}((x - \mu)/\sigma, -10, 10)$;
- Use $\text{KL}(\pi_{\text{old}}, \pi_{\text{new}})$ as diagnostic; spikes indicate a drop in performance.

7. When computing the gradient of an expectation, the parameter can be in the distribution, or in the integrand, or in both.

$$\begin{aligned}\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}} [f(X)] &= \mathbb{E} [f(x) \nabla_{\theta} \log p_{\theta}(x)] \\ \nabla_{\theta} \mathbb{E}_{x \sim N(0,1)} [f_{\theta}(x)] &= \mathbb{E} [\nabla_{\theta} f_{\theta}(x)]\end{aligned}$$

A **stochastic computation graph** is a DAG some of whose nodes are stochastic, *i.e.*, of the form “sample x from p_{input} ” instead of “ $x = f(\text{input})$ ”; we want the gradient of the expectation of the final output.

Examples include dropout or hard attention.

8. The **cross-entropy method** (CEM, aka CMA-ES) works well for Tetris (for a small number of parameters: 22). It is a generalization of the finite difference method, obtained with a 2-element population, defined with mirrored (antithetic) noise.

In practice, transform the reward $R(\tau)$ to its percentile or to an indicator variable for the best $k\%$ policies.

Derivative-free methods are not sample-efficient, but very easy to parallelize.

9. **Model-based reinforcement learning** is sample-efficient, and can even be used on real (physical) robots: model $p(s'|s, a)$ using Gaussian processes, neural nets of Gaussian model mixtures; optimize $\pi_{\theta}(a|s)$; iterate a few times (otherwise the distribution of states can be very different).

The global (DP, NN, GMM) can be used as a prior, with a local (Bayesian linear) model in the neighbourhood of the current trajectory, using a constraint

$$\text{KL}(p(\tau) \parallel p_{\text{old}}(\tau)) < \varepsilon.$$

For high-dimensional data (e.g., the goal could be specified by an image, with no explicit reward), learn in latent space.

10a. If an agent has *rational preferences* (*i.e.*, it cannot be taken advantage of), its preferences can be summarized by a **utility** function, and its behaviour described by *expected utility maximization* (von Neuman and Morgenstern theorem).

Humans are not rational (Allais’s paradox).

10b. **Inverse reinforcement learning** (IRL) infers a reward function from demonstrations – but it is not uniquely defined, and the demonstrations are not truly optimal. Do not learn a single reward function, but a distribution of reward functions – the maximum entropy one (the partition function is problematic, but the corresponding gradient descent algorithm can be seen as a generative adversarial network (GAN)).

11. The main challenges in RL are sample efficiency and exploration (we are still relying on random actions).

Distributional RL (C51) does not learn a value *function* but value *distributions*.

Unsupervised RL deals with sparse rewards by learning more about the environment through auxiliary prediction and control tasks (e.g., predicting that some pixels will change colours, or inducing those changes – for images, use CNN features instead of pixels).

Imagination-augmented agents learn an imperfect model, and use samples from that model as additional trajectories to learn from.

Hierarchical RL improves exploration by learning subtasks and combining them to reach the desired goal.

Feudal networks use two agents, a manager, who decides on the subtasks (and is rewarded if they help get close to the goal), and a worker, who tries to perform the subtasks (and is not aware of the ultimate goal).

For real robots, train the agent in a simulated environment, and fine-tune it in the physical world. **One-shot imitation learning** can then learn a new task from a single example.

12. Current reinforcement learning focuses on a single task, in contrast to machine learning, where generalization matters.

To improve generalization, train on different physical robots.

If the problem has some structure, if you can decompose the task into different “modules”, have separate neural nets learn them, and only combine them at the end (we already saw that with duelling DQNs).

Given a single demonstration, decompose it into intermediate steps (e.g., the latent representation from a CNN), and use them as intermediate goals. *Translation* from the demonstration context to the current context (e.g., the position of the objects) may be needed.

Model-based RL learns about the physical world and can use that knowledge to forecast the outcome of an action and act towards a new goal.

Multi-task learning is a form of regularization and data augmentation – some aspects of the world, useful to plan a task, may be easier to learn when performing another task.

Model-agnostic meta learning looks for a policy that can easily be adjusted to new tasks by policy gradient – for instance, a robot that could be steered in any direction after just one gradient step (*one-shot learning*).

13. The programming exercises use Chainer.

Can't decide? Undecide! C. Goodman-Strauss (2010)

Turing machines can be encoded in seemingly innocuous mathematical objects – the undecidability of the stopping problem leads to more concrete undecidable problems:

- Given a finite collection of tiles and a “seed” pattern, can we tile the whole plane?
- Collatz-like functions, in particular **Fractran** programs – given a list of fractions and a starting integer n , multiply it with the first fraction p/q in the list so that np/q is integer; continue with that integer – for instance

$$\frac{3}{11} \quad \frac{847}{45} \quad \frac{143}{6} \quad \frac{7}{3} \quad \frac{10}{91} \quad \frac{3}{7} \quad \frac{36}{325} \quad \frac{1}{2} \quad \frac{36}{6}$$

generates all prime powers of 10, in order;

- Post’s tag production systems: start with a sequence of zeroes and ones, remove the first three digits and add 00 or 1101 at the end, depending on whether the first digit was 0 or 1;
- SAT (with statements such as “the k th cell contains a at time t ”, “the machine is reading cell k at time t in state s ”, etc.
- Conway’s game of life;
- Wolfram’s rule 110.

Facets of entropy R.W. Yeung (2012)

The entropy function of a collection of discrete random variables $(X_i)_{1 \leq i \leq n}$ is

$$H : \begin{cases} \mathcal{P}([1, n]) & \rightarrow \mathbf{R} \\ \alpha & \mapsto H(X_\alpha) \end{cases}$$

where $H(X_\alpha)$ is the joint entropy of $(X_i)_{i \in \alpha}$. It satisfies the polymatroidal axioms

$$\begin{aligned} H(\emptyset) &= 0 \\ \alpha \subset \beta &\implies H(\alpha) \leq H(\beta) \\ H(\alpha) + H(\beta) &\geq H(\alpha \cap \beta) + H(\alpha \cup \beta). \end{aligned}$$

In addition to entropy, the *Shannon information mea-*

sures are

$$\begin{aligned} H(X|Y) &= H(X, Y) - H(Y) \\ I(X; Y) &= H(X) + H(Y) - H(X, Y) \\ I(X; Y|Z) &= H(X, Z) + H(Y, Z) - H(Z, Y, Z) - H(Z). \end{aligned}$$

Joint entropy and mutual information behave like set intersection and union.

$H(X)$	A
$H(Y)$	B
$H(X, Y)$	$A \cap B$
$I(X; Y)$	$A \cup B$

The polymatroidal axioms are equivalent to the basic inequalities:

$$\begin{aligned} \text{entropy} &\geq 0 \\ \text{conditional entropy} &\geq 0 \\ \text{mutual information} &\geq 0 \\ \text{conditional mutual information} &\geq 0 \end{aligned}$$

Those inequalities are not sufficient to describe entropy functions:

$$\begin{aligned} \mathcal{H}_n &= \{H : \mathcal{P}([1, n]) \rightarrow \mathbf{R}\} \\ \Gamma_n^* &= \{H \in \mathcal{H}_n : H \text{ is entropic}\} \\ \Gamma_n &= \{H \in \mathcal{H}_n : H \text{ polymatroidal}\} \\ \Gamma_n^* &\subset \Gamma_n \\ \Gamma_2^* &= \Gamma_2 \\ \Gamma_3^* &\subsetneq \Gamma_3 \text{ but } \overline{\Gamma_3^*} = \Gamma_3 \end{aligned}$$

For $n \geq 3$, Γ_n^* is neither closed nor convex, but $\overline{\Gamma_n^*}$ is a cone; for $n \geq 4$, there are many more non-Shannon-type inequalities: $\overline{\Gamma_4^*} \subsetneq \Gamma_4$.

Nonlinear time series analysis with R R. Huffaker et al. (2017)

Given a dynamical system, *i.e.*, a system of differential equations, we can study its orbits – they are sometimes chaotic, in the sense that

- Nearby trajectories diverge exponentially fast (chaos);
- The trajectories tend to accumulate around an “attractor” with non-integral dimension (strangeness).

For continuous (autonomous) systems, chaotic behaviour is only possible in dimension at least 3, but discrete systems (e.g., the *Poincaré map* of a continuous system, *i.e.*, the intersections of its trajectories with a plane) in dimension 1 or 2 can be chaotic.

Conversely, given just one coordinate of one of those trajectories, observed with noise, is it possible to reconstruct the other coordinates, the whole attractor, and the original dynamical system? This is the goal of nonlinear time series analysis.

The *embedding* of a univariate time series $(x_t)_{t \geq 0}$ with delay τ and dimension m is the multivariate time series $(x_t, x_{t+\tau}, \dots, x_{t+(m-1)\tau})_{t \geq 0}$. Under reasonable assumptions, it can be used to describe the dynamics of the system: one coordinate is indeed enough (Takens).

To estimate the delay τ , use the first minimum of the *automutual information* (AMI – but there is no theoretical justification, and even no guarantee that a minimum exists).

To estimate the embedding dimension, look at the proportion of *false near neighbours* (FNN), *i.e.*, the proportion of points close, in phase space, at time t , but not at time $t+1$ – sign that some information has been left out. PCA can also be used.

For this (and for the Lyapunov exponent), we want the points to be close in phase space, but not because they are close in time – they should be somehow “intependent”. The *Theiler window* specifies a minimum time separation. To estimate it, plot the proportion of FNN as a function of both space and time separation, or use the first zero of the autocorrelation function (ACF).

The dimension of the attractor can be defined using box counting

$$\lim_{\varepsilon \rightarrow 0} \frac{\log N(\varepsilon)}{\log(1/\varepsilon)}$$

where $N(\varepsilon)$ is the number of balls (or boxes) of size ε needed to cover the set or the *correlation dimension*

$$D_2 = \lim_{\varepsilon \rightarrow 0} \frac{\log C(\varepsilon)}{\log \varepsilon}$$

$$C(\varepsilon) = \lim_{n \rightarrow +\infty} \frac{1}{n^2} \# [|y_i - y_j| \leq \varepsilon]$$

(this is a different notion: it gives more weight to areas visited more often). The embedding dimension should be at least $2D + 1$.

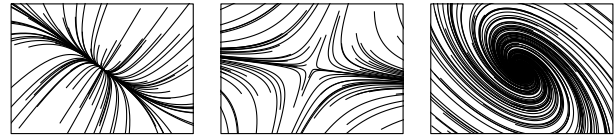
The *recurrence plot* shows the distance (in phase space) between the observation at time t and that at time s ; it can be thresholded to give a binary image, and reduced to a few numeric quantities: emphrecurrent quantitative analysis (RQA) looks, among other things, at the proportion of points in vertical or diagonal segments.

Singular spectrum analysis (SSA) can help separate signal from noise – but it is manual, and some of the components have to be grouped.

To test for stationarity, use Schreiber’s nonlinear cross-prediction stationarity test (divide the time series into non-overlapping segments; fit a nonlinear model on each, *e.g.*, k -NN on an embedding; measure the performance of model i on segment $j \neq i$ versus that on segment i).

A linear system is a differential equation of the form $\dot{y} = Ay$. With real eigenvalues, there can be a stable, unstable or saddle point; with complex eigenvalues, it can be a stable or unstable focus equilibrium, or a centre point. If you see exponentially damped or

exploding trajectories, use a linear model.



To test for nonlinearity, use the BDS test, or the *Hellinger distance* $S_p(k)$ between $f_{X_t, X_{t+k}}$ and $f_{X_t} f_{X_{t+k}}$ (which can be seen as a “nonlinear correlation”) or the moving block bootstrap (in a time series of 0s and 1s, estimate \hat{p} and check if its variance is $\sqrt{\hat{p}(1-\hat{p})/n}$ as it would for an iid sequence).

Non-linearity can also be assessed using *surrogate data*, *i.e.*, time series with the same “linear properties”, *e.g.*,

- ARMA surrogates: Fourier-transform the time series, randomize the phases, transform it back;
- Amplitude-adjusted Fourier transforms (AAFT) surrogates: idem, but transform the output to recover the same distribution;
- PPS surrogates (for aperiodic oscillations): (first coordinate of a) random walk on the reconstructed shadow attractor

and looking at some statistic, *e.g.*, correlation dimension, maximum Lyapunov exponent, nonlinear prediction error or permutation entropy.

Granger causality only works for stochastic systems. For deterministic ones, try to predict future values of y using x ’s attractor (convergent cross-mapping, CMM): if $x(t_a), x(t_b), \dots$ are neighbours of $x(t_0)$, then $y(t_a), y(t_b), \dots$ should be close to $y(t_0)$, and increasingly so as more data arrives. Delayed cross-mapping check how the forecast skill changes with the lag between x and y .

To detect changepoints, compute the SSA decomposition before and after a possible changepoint, compute the scalar product (*i.e.*, correlation) between the first eigenvectors, and compare with surrogate data to have a p -value.

To detect changepoints, one can also estimate the distribution of values on a moving window: it will usually have one peak, but may have two (or be a mixture distribution) during transition periods

Phenomenological modeling estimates the coefficients of an ODE from data, *e.g.*, with regression, assuming the coefficients are polynomial, approximating the derivatives with high (4th) order centered finite differences, with the lasso or adaptive lasso ($|\beta|^\nu$ penalty, $0 < \nu < 1$) to deal with multicollinearity.

Implementations in `tseriesChaos` (`mutual`, `d2`, `false.nearest`, `lyap_k`), `nonlinearTseries` (RQA), `tseriesEntropy` (`Srho.test.ts`, `surrogateAR`, `Trho.test.AR`, `Trho.test.SA`), `Rssa`, `fractal` (`surrogate`), `multispatialCCM`, `MESS`, `extRemes`.

**A differential equation for modeling
Nesterov’s accelerated gradient method:**

Nesterov momentum

$$\begin{aligned}x_k &= y_{k-1} - s \nabla f(y_{k-1}) \\ y_k &= x_k + \frac{k-1}{k+2}(x_k - x_{k-1})\end{aligned}$$

by a can be modeled by an ODE

$$\ddot{x} + \frac{3}{t}\dot{x} + \nabla f(x) = 0.$$

The $O(1/t^2)$ convergence rate is guaranteed if the damping term $\frac{3}{t}\dot{x}$ is sufficiently large: 3 is the smallest constant guaranteeing it but, for large t , the system is over-damped. This suggests resetting t whenever $\langle \dot{x}, \ddot{x} \rangle = 0$ or, for Nesterov momentum, resetting k whenever Δx starts decreasing.

Learning gradient descent: better generalization and longer horizons K. Lv et al. (2017)

The optimization algorithm can be added to the model, to have the computer choose the best stochastic gradient descent variant. For instance, one can use the gradient as input and the updates as output of a coordinate-wise LSTM cell whose latent variables are the first and second moments of the previous gradients (momentum and AdaDelta’s normalizing factors). To help make the algorithm scale-invariant:

- Randomly scale the functions to minimize at training time;
- Add a random convex function, $g(x) = \frac{1}{n} \sum (x_i - v_i)^2$ to make the objective better-behaved;
- Do not feed the raw gradient and momentum to the LSTM, but rescale them with the AdaDelta factor.

Fighting biases with dynamic boosting A.V. Dorogush et al.

Boosting uses the same model to build the data and to compute the gradient: this data reuse introduces bias in the gradient. CatBoost avoids it.

Oblivious decision trees use the same criterion across an entire level of the tree.

Escaping from saddle points – online stochastic gradient for tensor decomposition R. Ge et al.

Even for non-convex functions, SGD converges to a local minimum in a polynomial number of iterations.

Failures of gradient-based deep learning S. Shalev-Shwartz et al

Gradient-based optimization fails when:

- The gradient is too flat and/nor not informative enough about the position of the minimum;

- The sample gradient has a low signal-to-noise ratio;
- The condition number is high.

Possible solutions include:

- Preconditioning;
- Intermediate losses in end-to-end architectures;
- “Forward-only update rule”: if a derivative is zero, in intermediate computations, replace it with 1.

An overview of gradient descent optimization algorithms S. Ruder

A tutorial on Bayesian optimization of expensive cost functions, with applications to active user modeling and hierarchical reinforcement learning R. Brochu et al. (2010)

Illustrated introduction to Bayesian optimization followed by non-trivial examples:

- Bayesian modeling of preferences

$$\begin{aligned}P[X \succ Y] &= \Phi(f(X) - f(Y)) \\ f &\sim \text{GP}\end{aligned}$$

- (the Bradley–Terry model assumes f is linear and uses a logit instead of the probit Φ);
- Hierarchical reinforcement learning.

Batched high-dimensional Bayesian optimization via structural kernel learning Z. Wang et al. (2017)

Bayesian optimization can scale to high-dimensional functions by assuming a latent additive structure, and learning it with Gibbs sampling (Dirichlet prior for the mixing proportions: $\theta \sim \text{Dir}(\alpha)$, $z_j \sim \text{Multi}(\theta)$). Function evaluations can be batched using a determinantal point process (DPP) and processed in parallel.

Distribution-free predictive inference for regression J. Lei et al. (2017)

Conformal inference provides distribution-free, finite-sample prediction sets. With a naive approach, fitting the model on $(x_1, y_1), \dots, (x_n, y_n)$ and using the quantiles of the residuals to form a prediction interval $\hat{\mu}(x_{n+1}) \pm q_\alpha$, the prediction intervals are too small.

Instead, fit the model on $(x_n, y_1) \dots, (x_n, y_n), (x_{n+1}, y)$, for all values of y : the prediction set is the set of y ’s in their $\hat{\mu}(x_{n+1}) \pm q_{y,\alpha}$ interval.

To reduce the computations, do not refit the model for all values of y , but split the data into equal-sized parts, use the first to fit the model, and the second to estimate the residual quantiles. Using N splits, each at level α/N , and taking the intersection of the corresponding intervals, reduces randomness but enlarges the interval – the Bonferroni effect dominates. The

Jackknife (using quantiles of the leave-out-one residuals) relies on a similar idea, but is more fragile, unless we impose strong conditions on the estimator.

**Multivariate quantiles
and multivariate L-moments
A. Decurving (2014)**

Univariate **L-moments** can be defined from order statistics:

$$\begin{aligned}\lambda_1 &= E[X] \\ \lambda_2 &= \frac{1}{2}E[X_{(2)} - X_{(1)}] \\ \lambda_3 &= \frac{1}{3}E[X_{(3)} - 2X_{(2)} + X_{(1)}] \\ &\dots \\ \lambda_r &= \frac{1}{r} \sum (-1)^r \binom{r-1}{k} E[X_{r-k:k}]\end{aligned}$$

or by projecting the quantile function Q on the shifted Legendre polynomials (an orthogonal basis of $L^2([0, 1]; \mathbf{R})$ for the scalar product $\langle f, g \rangle = \int_0^1 fg$ – Legendre polynomials use $\int_{-1}^1 fg$).

Multivariate L-moments can be defined using:

- The monotone transport between the distribution of interest and $\text{Unif}[0, 1]^d$ to define the quantiles, $Q(F(u)) = T(u)$;
- The basis of products of univariate Legendre polynomials.

Other target distributions (the min-copula, a standard Gaussian) and other transports (e.g., Rosenblatt), or other orthogonal polynomials (Hermite) can be used.

The **monotone transport** from μ to ν is the only measurable map T such that $T_{\#}\mu = \nu$ and $T = \nabla\phi$, for some convex function ϕ (it is guaranteed to exist if $\mu = \text{Unif}[0, 1]^d$ or, more generally, if it gives zero measure to sets of Hausdorff measure at most $d - 1$) [Brenier’s theorem].

The **Rosenblatt transport** is

$$T(x_1, \dots, x_d) = \begin{pmatrix} F_{X_1}(x_1) \\ F_{X_2|X_1=x_1}(x_2) \\ F_{X_3|X_1=x_1, X_2=x_2}(x_3) \\ \vdots \end{pmatrix}.$$

The **power Voronoi diagram** is defined using the “distance”

$$d(x_i, u) = \|u - x_i\|^2 + w_i.$$

The convex piecewise linear function

$$\phi_h(u) = \text{Max}_i u \cdot x_i + h_i,$$

for $h_i = -\frac{1}{2}(\|x_i\|^2 + w_i)$ has constant gradient x_i on each Voronoi cell. The monotone transport between μ and the empirical distribution of a sample x_1, \dots, x_n is $T = \nabla\phi_h$ with

$$h = \text{Argmin} \int \phi_h(u) du - \frac{1}{n} \sum h_i$$

(which can be computed by Newton’s method).

**Forecaster’s dilemma:
extreme events and forecast evaluation
S. Lerch et al.**

Forecasts are often only evaluated when an extreme event occurs (earthquake, financial crisis, etc.): this encourages forecasters to always predict disaster – whenever they will be examined, they will be right. There is no fix for point forecasts, but probability forecasts are more flexible. The joint distribution of forecast F and observation Y can be decomposed as

$$[F, Y] = [F][Y|F] = \text{sharpness} \times \text{calibration}.$$

The quality of a forecast distribution F for an observation y can be measured by the logarithmic score

$$\text{LogS}(F, y) = -\log f(y)$$

or by the continuously ranked probability score

$$\text{CRPS}(F, y) = \int_{-\infty}^{+\infty} (F(y) - \mathbf{1}_{y \leq z})^2 dz.$$

There are weighted variants:

$$\text{CL}(F, y) = -w(y) \log \frac{f(y)}{\int w f}$$

$$\text{twCRPS}(F, y) = \int w(z) (F(z) - \mathbf{1}_{y \leq z})^2 dz.$$

**The Cramér distance
as a solution to biased Wasserstein gradients
M.G. Bellemare et al.**

In machine learning, the Kullback-Leibler divergence (relative entropy) is often used to measure the “distance” between the data and the fitted model. Instead, one can use the *Wasserstein metric*, which does not only consider probabilities, but also proximities.

It is scale-invariant

$$d(cX, cY) \leq |c|^\beta d(X, Y)$$

and sum-invariant

$$d(A + X, A + Y) = d(X, Y)$$

but does not have unbiased sample gradients

$$\nabla_\theta d(P, Q_\theta) \neq \mathbb{E}_{x_1, \dots, x_m \sim p} \nabla_\theta d\left(\sum \delta_{x_i}, Q_\theta\right).$$

Prefer the **Cramér distance**

$$d_2(P, Q) = \left(\int (F_P(x) - F_Q(x))^2 dx \right)^{\frac{1}{2}},$$

which is scale-invariant, sum-invariant, and has unbiased sample gradients.

A note on the evaluation of generative models
L. Theis et al. (2016)

The *Jensen-Shanon divergence* (JSD) is a symmetrized KL divergence:

$$\text{JSD}(p, q) = \frac{1}{2} \text{KL}\left(p \parallel \frac{p+q}{2}\right) + \frac{1}{2} \text{KL}\left(q \parallel \frac{p+q}{2}\right)$$

The *maximum mean discrepancy* (MMD) is

$$\text{MMD}(p, q) = \mathbb{E}_{\substack{x, x' \sim p \\ y, y' \sim q}} [k(x, x') - 2k(x, y) + k(y, y')]^{1/2}.$$

Random projection through multiple optical scattering: approximating kernels at the speed of light
A. Saade et al.

Analogue, optical devices can efficiently compute random projections.

Alpha-beta divergences discover micro and macro structures in data
K. Narayan et al. (2015)

The Kullback-Leibler divergence

$$D(P\|Q) = \sum_{i \neq j} P_{ij} \log \frac{P_{ij}}{Q_{ij}}$$

minimized by t-SNE can be replaced by the α - β -divergence

$$D(P\|Q) = \frac{1}{\alpha\beta} \sum_{i \neq j} -P_{ij}^\alpha Q_{ij}^\beta + \frac{\alpha}{\alpha+\beta} P_{ij}^{\alpha+\beta} + \frac{\beta}{\alpha+\beta} Q_{ij}^{\alpha+\beta}.$$

Different values of (α, β) focus on macro-structures, micro-structures or hard-to-classify observations ($\alpha \approx 1, \beta \approx 0, \alpha + \beta \approx 1$).

Entropic graph-based posterior regularization
M.W. Libbrecht et al. (2015)

To encourage variables associated to nodes in a graph to have similar posterior distributions when those nodes are linked, add a penalty for their KL divergence. More precisely, *posterior regularization* introduces an auxiliary joint distribution q , adds a regularizer to it, and a penalty to make it close to the posterior p ,

$$\text{Penalty}(p) = \max_q -D(q\|p) + \text{Penalty}(q).$$

Persistence topology of syntax
A. Port et al.

Persistence homology shows that language evolution cannot be summarized by a tree, but often needs a more general phylogenetic *network* – for Indo-European, the first homology generator comes from ancient Greek.

Spin class models of syntax and language evolution
K. Siva et al.

Spin model to model and forecast the evolution of a weighted graph with binary feature vectors at each node.

Graph grammars, insertion Lie algebras and quantum field theory
M. Marcolli and A. Port

Graph grammars generalize context-free and context-sensitive grammars, and model parallelism: the production rules replace a subgraph (a single, non-terminal node, in the case of context-free grammars) with a new graph.

Prevalence and recoverability of syntactic parameters in sparse distributed memories
J.J. Park et al.

A *Kanerva network* (or *sparse distributed memory*) learns a mapping $\mathbf{F}_2^N \rightarrow \{\pm 1\}$ from a dataset $(x_i, y_i)_i$ as follows:

- Pick k points $\hat{x}_1, \dots, \hat{x}_k \in \mathbf{F}_2^N$, each associated with a count variable $\hat{y}_1, \dots, \hat{y}_k$, initialized at 0;
- For each observation (x_i, y_i) , find the points within distance d of x_i , and increment/decrement their counts.

Here, they are used to check which language features (SVO order, etc. – from the SSWL (now TerraLing) or WALS databases) can be recovered. They can also learn the identity map $\mathbf{F}_2^N \rightarrow \{\pm 1\}^N$, *i.e.*, learn a dataset.

Implementation in `msbrogli/sdm`.

Submodularity in data subset selection and active learning
K. Wei et al. (2015)

To select a small sample of data on which to train a classifier with minimal performance loss, consider the data log-likelihood,

$$\ell(S) = \sum_{i \in V} \log p(x_i, y_i | \theta(s))$$

i.e., the likelihood of the whole data V when the model is estimated on a subset $S \subset V$. For the naive Bayes or the nearest neighbour classifiers, this is a difference of submodular functions but, under the constraints $|S| = k$ and S balanced (same label distribution as V), it reduces to a modular function, which can be approximately maximized with the lazy greedy algorithm.

***Langevin diffusions and
the Metropolis-adjusted Langevin algorithm***
T. Xifara et al.

Given a diffusion $dX_t = \mu(X_t)dt + \sigma(X_t)dW_t$, the probability density function $p(x, y)$ satisfies the Fokker-Plank equation

$$\frac{\partial p}{\partial t} = -\frac{\partial(\mu p)}{\partial x} + \frac{1}{2} \frac{\partial(\sigma^2 p)}{\partial x^2}$$

or

$$\frac{\partial p}{\partial t} = -\mathbf{1}' \nabla_x (\mu p) + \frac{1}{2} \mathbf{1}' \nabla_x^2 (\sigma \sigma' p) \mathbf{1}$$

in dimension n . One can therefore build a diffusion with a prescribed stationary distribution, e.g.,

$$dX_t = \frac{1}{2} \nabla \log \pi dt + dW_t$$

or

$$dX_t = \frac{1}{2} A \nabla \log \pi dt + \Gamma dt + \sqrt{A} dW_t$$

(with $\Gamma(x) = 0$ if $A(x)$ does not depend on x).

***FairTest: discovering unwarranted
associations in data-driven applications***
F. Tramèr et al.

Unwarranted associations are statistically significant associations, in a subpopulation, between a protected attribute and an output, with no accompanying explanatory factor. Examples include

- Unintended side effects (e.g., discounted prices if there is a competitor nearly exclude low-income areas);
- Large errors affecting a subpopulation (e.g., future health prediction for the elderly).

***The dual-sparse topic model: mining focused
topics and focused terms in short text***
T. Lin et al. (2014)

Spike-and-slab prior for sparse topic mixtures and sparse word usage.

***A unified model
for unsupervised opinion spamming detection
incorporating text generality***
Y. Xu et al.

Hierarchical Bayesian model for spam detection, combining text, user and item features, estimated with Gibbs-EM, *i.e.*, alternating between collapsed Gibbs sampling and variational inference gradient descent.

***Bayesian post-selection inference
in the linear model***
S. Panigrahi et al.

In *selective inference* (adaptive data analysis), the analyst looks at the data before deciding which question to ask. This can be modeled by *conditioning* on selection, *i.e.*, by using a *truncated* log-likelihood, for a sequence

model (k largest statistics, BY correction) or a (generalized) linear model (through a convex approximation of the truncated likelihood).

***Learning the nonlinear geometry of
high-dimensional data: models and algorithms***
T. Wu and W.U. Bajwa (2015)

Union-of-subspace (UoS) models have trouble when the subspaces are close: constrain the subspaces to be close, using a distance on the Grassmanian, e.g., $d(S_1, S_2) = \|D_1 - P_{S_2} D_1\|_F$ where D_i is an orthonormal basis of S_i and P_{S_2} is the projection on S_2 – it can also be defined from the *principal angles* θ^k as $d(S_1, S_2) = \sqrt{s - \sum \cos^2 \theta_{12}^k}$, where s is the subspace dimension (for nonlinear data, use a kernel).

***Scalable Gaussian processes for characterizing
multidimensional change surfaces***
W. Herlands et al. (2015)

Mixture models

$$f(x) = \sum p_i f_i(x), \quad \mathbf{p} = \text{softmax}(\mathbf{w}(x)),$$

with a Gaussian process prior on \mathbf{w} , can be seen as generalizations of changepoint models.

Deterministic independent component analysis
R. Huang et al.

Variants of the HKICA algorithm (FastICA, *i.e.*, finding directions maximizing some measure of non-Gaussianity, only has theoretical guarantees in the noiseless case):

- Sample $\phi, \psi \sim N(0, I)$;
- Compute $m_p(\eta) = E[(\eta' X)^p]$, $f = \frac{1}{12}(m_4 - 3m_2^2)$, $\nabla^2 \hat{f}$;
- Compute the eigenvectors of $(\nabla^2 \hat{f} \phi)(\nabla^2 \hat{f} \psi)^{-1}$, $A = (\mu_1 | \dots | \mu_d)$;
- We then have $\mathbf{x} \approx A\mathbf{s} + \text{Gaussian noise}$.

On restricted nonnegative matrix factorization
D. Chistikov et al. (2016)

The *nonnegative rank* of an $n \times m$ nonnegative matrix M is the smallest d for which we can find nonnegative $n \times d$ and $d \times m$ matrices W and H such that $M = WH$. If M is rational, W and H need not be so, *i.e.*, $\text{rank}_+^{\mathbf{Q}} > \text{rank}_+^{\mathbf{R}}$ in general (in dimensions beyond 3). The *restricted NMF* requires $\text{rank } M = \text{rank } W$, defining a *restricted nonnegative rank*.

***Distributional rank aggregation
and an axiomatic analysis***
A. Prasad et al. (2015)

Rank aggregation is the problem of combining several rankings (e.g., from search engines) into one. Distributional rank aggregation only uses the distribution (histogram) of the rankings. The normative axioms of “social welfare theory”, non-dictatorship, universality,

transversality, Pareto efficiency and independence to irrelevant alternatives (Arrow's impossibility theorem) can be relaxed and satisfied.

Parallel resampling in the particle filter
L.M. Murray et al. (2015)

The propagation and weighting steps of sequential Monte Carlo (SMC, particle filters) are easy to parallelize, but the resampling step is less so – try:

- Rejection sampling, if an upper bound on the weights is known:

$$j \sim \text{Unif}[1, N] \text{ while } \text{Unif}(0, 1) > w_1/w_{\max};$$

- Metropolis: run N identical Markov chains in parallel, for B steps, sampling from $\text{Multinom}(\mathbf{w})$ – only the weight ratios w_i/w_j are needed.

Scalable nearest neighbor algorithms for high-dimensional data
M. Muja and D.G. Lowe (2014)

The FLANN library for approximate nearest neighbour search automatically selects the algorithm among:

- *Randomized k -d forests*: multiple k -d trees searched in parallel, in which the split dimension is chosen randomly from the top 5 with highest variances – non-axis-aligned variants exist but do not perform significantly better;
- *Priority search k -means trees*: find k clusters and process each cluster recursively until they reach a minimum size – it is just another way of hierarchically partitioning the space.

Six myths of polynomial interpolation quadrature
L.N. Trefethen

Equispaced interpolation may diverge (Runge phenomenon) but *Chebyshev interpolation*, i.e., interpolation on $[-1, 1]$ at $x_k = \cos(j\pi/n)$, converges for Lipschitz continuous functions.

It can be numerically evaluated with the *barycentric*

formula

$$\begin{aligned}\ell_j(x) &= \prod_{i \neq j} \frac{x - x_i}{x_j - x_i} \\ \ell(x) &= \prod (x - x_i) \\ \ell'(x_j) &= \prod_{i \neq j} (x_i - x_j) \\ \ell_j(x) &= \frac{\ell(x)}{x - x_j} \frac{1}{\ell'(x_j)} = \frac{\ell(x)}{x - x_j} w_j \\ g(x) &= \sum \ell_j(x) y_j = \ell(x) \sum \frac{w_j}{x - x_j} y_j \\ 1 &= \ell(x) \sum \frac{w_j}{x - x_j} \\ g(x) &= \frac{g(x)}{1} = \frac{\sum \frac{w_j}{x - x_j} y_i}{\sum \frac{w_j}{x - x_j}}\end{aligned}$$

The monomials x^k are well-suited to find roots on the circle; for roots on $[-1, 1]$, prefer Chebyshev polynomials.

Fixed points of belief propagation an analysis via polynomial continuation
C. Knoll et al.

Homotopy continuation is a way of numerically finding the solutions of a system of equations, by following a homotopy from an easy-to-solve system to the desired one. In the case of a polynomial system, the *polyhedral homotopy method* finds all the solutions (it scales better than Gröbner bases).

Non-free implementation in Hom4PS-3.

Transfinite game values in infinite chess
C.D.A. Evans and J.D. Hamkins

A position has value (at most) n if there is a strategy leading to a check-mate in (at most) n moves, whatever the opponent does.

A position with value ω , the first infinite ordinal, is a position with Black to play resulting in a mate-in- n position for White, with n as large as Black wants.

A position with value $\omega + n$ is n moves away from a position with value ω .

A position with value $n\omega$ is a position in which Black can announce n times “I will make an announcement in k_i moves” or, for the last one, “I will lose in k_n moves”, with the k_i 's as large as desired.

In a position with value $k\omega$, Black can play to be in a position with value $k\omega$, with k as large as desired.

A position in infinite chess with game value ω^4
C.D.A. Evans et al.

What is a Leavitt path algebra?

G. Abrams

Given a monoid M , it is possible to find a ring R so that $(\text{Mod } R, \oplus) \simeq M$.

Highly comparative time-series analysis: the empirical structure of time series and their methods

B.F. Fulcher et al. (2013)

The `hctsa` Matlab library (GPL) provides 1000 time series features for classification, clustering and biclustering, to be used with the UCR time series dataset for nearest neighbour search.

FATS: Feature analysis for time series

I. Nun et al. (2015)

The FATS Python library computes 30 time series features (to classify astronomy time series).

Feature-based time series analysis

B.D. Fulcher (2017)

Look at global features:

- Distribution of the values (μ , σ , etc.);
- Stationarity, e.g., $\text{StatAv} = \frac{\text{sd}(\bar{x}_{1:w}, \bar{x}_{w+1:2w}, \dots)}{\text{sd}(x)}$
- ACF, Fourier and wavelet coefficients;
- Nonlinear time series analysis: Lyapunov coefficients, correlation dimension, correlation entropy;
- Entropy: approximate entropy, sample entropy, permutation entropy;
- Scaling (fractality): Hurst exponent (DFA);
- Time series models: parameters and goodness-of-fit statistics, properties of the residuals;

and subsequence features:

- If the time series are aligned, consider $\text{Mean}(x_{i:j})$, $\text{Sd}(x_{i:j})$, $\text{Slope}(x_{i:j})$ – these are families of features: select them greedily, using the entropy gain (as in decision trees);
- *Shapelets* are small patterns that may appear in a time series,

$$\min_i d(x_{i-w:i+w}, \text{shapelet})$$

selected by genetic algorithms – you can also look at their position or the number of occurrences.

Automatic time series phenotyping using massive feature extraction

B.D. Fulcher and N.S. Jones (2016)

Normalize the time series with a “robust sigmoid”,

$$x \leftarrow \left(1 + \exp - \frac{x - \text{median}}{1.35 \times \text{IQR}} \right)^{-1}$$

Large-scale unusual time series detection

R.J. Hyndman et al. (2015)

A few more time series features:

- Lumpiness: variance of the variances in 24-observation blocks;
- Spikiness: variance of the l.o.o. variances of the STL residuals;
- Level and variance change: maximum change in mean or variance in consecutive 24-observation blocks;
- KL score: maximum Kulback-Leibler divergence between kernel density estimates of consecutive 48-observation blocks;
- Flat spots: maximum run length of the (10-value) quantized time series.

Some features are not defined: peak, trough, curvature.

Forecastable component analysis

G.M. Goerg (2013)

Find orthogonal directions, maximizing *forecastability*, measured by $\Omega = 1 - H / \log 2\pi$, where H is the **spectral entropy**, *i.e.*, the entropy of the spectral density (the spectrum, rescaled to be a probability density on \mathbf{S}^1). Implementation in `ForeCA::Omega`.

A scalable method for time series clustering

X. Wang et al. (2004)

Cluster time series using the following features:

- Trend = $1 - \frac{\text{Var}(\text{detrended})}{\text{Var}(\text{raw})}$
- Seasonality = $1 - \frac{\text{Var}(\text{deseasoned})}{\text{Var}(\text{raw})}$
- Periodicity: position of the first peak in the ACF, if it is positive, and significantly higher than the previous trough;
- Serial correlation: Box-Pierce statistic;
- Non-linearity: Teräsvirta test statistic;
- Skewness, kurtosis, Hurst exponent, maximum Lyapunov exponent.

All those features have values in $[0, +\infty)$: rescale them parametrically

$$q \mapsto \frac{e^{aq} - 1}{b + e^{aq}}$$

so that quantiles 10% and 90% correspond to time series with and without the features (respectively an example from the literature, and white noise).

TSclust: an R package for time series clustering

P. Montero and J.A. Vilar (2014)

The TSclust package defines the following dissimilarity measures:

- L^p distance;
- Dynamic time warp (DTW) distance,

$$d(x, y) = \min_r \sum_{(i,j) \in r} |x_i - y_j|;$$

- Fréchet distance,

$$d(x, y) = \min_r \max_{(i,j) \in r} |x_i - y_j|;$$

- Correlation:

$$\sqrt{2(1 - \text{Cor}(x, y))}, \quad \left(\frac{1 - \text{Cor}(x, y)}{1 + \text{Cor}(x, y)} \right)^{\beta/2};$$

- L^2 distance between the ACF, with constant or exponentially decaying weights;
- L^2 distance between the periodograms, the normalized periodograms, the integrated periodograms (Cramér–von Mises distance between the spectral densities);
- Distance between the wavelet coefficients, the $AR(\infty)$ coefficients or the cepstral coefficients;
- Statistics from ARMA tests checking if the two time series come from the same model;
- Distance between the SAX representations (obtained by aggregating, and then quantizing with quantiles);
- Divergence between the distributions of the permutations induced by an m -dimensional embedding;
- Normalized compression distance and variants:

$$\frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}, \quad \frac{C(xy)}{C(x)C(y)};$$

- Distance between the k -step-ahead forecast distributions, from a *sieve bootstrap* (i.e., bootstrap on the residuals of an AR or similar model).

Those measures can be corrected, for correlation or complexity, by multiplying by

$$\frac{1}{1 + \exp k \text{Cor}(\Delta x, \Delta y)} \quad \text{or} \quad \frac{\max\{\text{CE}(x), \text{CE}(y)\}}{\min\{\text{CE}(x), \text{CE}(y)\}},$$

where $\text{CE}(x) = \|\Delta x\|_2^2$.

***pdc: an R package
for complexity-based clustering of time series***
A.M. Brandmaier (2015)

To cluster time series, use an m -dimensional embedding, replace the m -dimensional vectors with the corresponding permutation (their ranks), and use the *Hellinger distance* (a metric approximation of the KL divergence) between the permutation distributions as a dissimilarity measure. Choose the embedding dimension m and the delay τ to minimize the (average) normalized *permutation entropy*

$$-\frac{1}{m!} \sum_{\sigma \in \mathfrak{S}_m} p_\sigma \log p_\sigma.$$

***Visualising forecasting algorithm performance
using time series instance spaces***
Y. Kang et al. (2016)

Time series features (spectral entropy, trend and seasonality strength from the STL decomposition, period (if known), ACF, optimal Box-Cox transformation to make the time series linear) can be used to

- Check how diverse the M3 dataset (in the MComp package) is;
- Add synthetic time series in its gaps, using a genetic algorithm;
- Visualize which forecasting algorithms perform well with which type of time series.

***JIDT: an information-theoretic toolkit
for studying the dynamics of complex systems***
J.T. Lizier (2014)

The JIDT Java library provides several information-theoretic measures for time series, centered on *transfer entropy*, and several estimators.

Entropy rate	$H(X_{n+1} X_n)$
Active information storage	$I(X_n; X_{n+1})$
Transfer entropy	$I(Y_{n+1}; X_{n+1} X_n)$
Conditional TE	$I(Y_{n+1}; X_{n+1} X_n, Z_n)$

(you can replace X_n with (X_n, \dots, X_{n+k-1})).

Multiscale entropy analysis
M. Costa et al. (2000)

Compute the average of the datapoints on non-overlapping windows of size τ , and then the sample entropy; plot the sample entropy versus τ – for $1/f$ noise, it is flat.

***Introducing nonlinear time series analysis
in undergraduate courses***
M. Perc (2004)

Nonlinear time series analysis studies stationary, deterministic time series, from data alone, i.e., without knowledge of the underlying dynamical system.

The dynamical system can often be recovered from the *Takens embedding*, i.e., the cloud of points $(x_t, x_{t+\tau}, \dots, x_{t+(m-1)\tau}) \in \mathbf{R}^m$. To choose the delay τ , check when the ACF decreases to $1/e$, or when the AMI (auto-mutual information) reaches its first minimum – though there is no reason such a minimum should exist. To choose the embedding dimension m , notice that two points close in the embedding should remain close at the next time step – if not, i.e., if

$$\|p_t - p_s\| \leq \sigma \quad \text{but} \quad \frac{|x_{t+m\tau} - x_{s+m\tau}|}{\|p_t - p_s\|} > 10,$$

they are *false nearest neighbours* (FNN) – we are missing some information about them. Plot the proportion of FNN as a function of m to help select m .

To test for stationarity, split the time series into non-overlapping segments and compare the mean and standard deviation on each segment, or look at the *cross-prediction error* δ_{ij} when forecasting the next observation in segment j with a k -NN model fitted on segment i .

To recover the dynamical system,

- Split the embedding space into boxes;
- Whenever a trajectory crosses a box, compute its average direction, as a unit vector;
- Average the vectors in each (non-empty) box – if the directions are consistent, the average will still have unit norm.

The *Lyapunov exponent* measures how fast nearby trajectories diverge; it can be estimated by choosing a point p_s close to p_t and averaging

$$\frac{1}{\nu} \log \frac{\|p_{t+\nu} - p_{s+\nu}\|}{\|p_t - p_s\|}$$

with $\nu = m\tau$, and replacing the point p_s at each step, trying to keep the same direction.

Nonlinear time series analysis revisited **E. Bradley and H. Kantz (2015)**

Nonlinear time series analysis reconstructs the state space of a dynamical system, from data alone, to compute fractal dimension, Lyapunov exponents (instability), Kolmogorov-Sinai entropy (unpredictability), etc. – but noise and the finiteness of the data call for caution.

- The fraction of pairs of data points at distance at most ε (the *correlation sum*) scales as ε^{D_2} , where D_2 is the *fractal (Renyi) dimension*.
- The time series should be long enough: as a rule of thumb, $N > 42^{D_2}$.
- To test if the values are significantly different from a null model, use *surrogate time series*: Fourier transform the data, randomize the phases, transform back, map onto the original values (only keeping the rank), and somehow recover the correlations.
- Permutation entropy.
- Recurrence plots and *recurrence quantitative analysis* (RQA): proportion of black points in the plot, in diagonals, in vertical lines, length distribution of those lines.
- Network characteristics: interpret the recurrence matrix as a graph and look at some of its metrics: centrality, shortest paths, clustering coefficients, etc.

Distinguishing noise from chaos is difficult.

Complex network approach for recurrence analysis of time series **N. Marwan et al. (2009)**

The following five RQA features

- Maximum diagonal length;
- Laminicity: proportion of points in vertical lines;
- Link density (for the network whose adjacency matrix is the recurrence plot);
- Clustering coefficient;
- Average minimum path length

can identify different regimes of the logistic map $x_{n+1} \leftarrow \alpha s_n(1 - x_n)$ as α varies, or (when estimated on a moving window) the different regimes of a real-world time series.

Characterizing the structural diversity of complex networks across domains **K. Ikehara and A. Clauset (2016)**

To compare networks, look at scale-invariant features: clustering coefficient, degree assortativity and network motifs (compared with a random graph with the same degree distribution).

Time series classification with COTE: the collective of transformation-based ensembles **A. Bagnall et al. (2014)**

Classify time series using ensemble methods (random forests or **rotation forests** – random forests with a PCA on a random subset of the variables at each node), on data transformed into the time (shapelets), frequency (periodogram) or autocorrelation (ACF, PACF, AR coefficients) domains.

The **shapelet features** are the distances to smaller time series s (“shapelets”),

$$d(s, x) = \min_{\substack{y \subset x \\ \text{len}(x) = \text{len}(s)}} d(s, y);$$

those smaller time series are taken from subsequences of the training data, pruned to retain only the most discriminating ones (small distances to one class, large distance to the others) and to avoid redundancy.

Forecasting at scale **S.J. Taylor and B. Letham**

To forecast time series, Facebook uses a GAM-like model, with trend, seasonality (Fourier) and (irregular) holiday components – curve fitting is easier than time series modeling. The trend is either piecewise linear (with a large number of potential changepoints, but a Laplace prior keeps them sparse), or a logistic growth model, with changepoints accounting for changes in market size (e.g., using the population from the World Bank).

R/Python implementation, via Stan, in `fbprophet`.

Distributed and parallel time series feature extraction for industrial big data applications **M. Christ et al. (2017)**

Select time series features (from `hctsa`) using tests for feature \perp outcome (Fisher, Kolmogorov-Smirnov or Kendall rank, depending on whether the variables are binary or continuous) and the Benjamini-Yekutieli (BY) procedure to control the false discovery rate (FDR).

The all-relevant feature selection using random forest **M.B. Kursu and W.R. Rudnicki (2010)**

To find all the relevant features (instead of just a non-redundant set), add a “shadow” variable for each feature, by shuffling its values, fit a random forest, and

mark as relevant the features more important than the best random one; remove them, and iterate until nothing left is relevant.

R implementation in Boruta.

Surrogate time series

T. Schreiber and A. Schmitz (1999)

To test if a time series is linear, e.g.,

- H_0 : iid;
- H_0 : ARMA (“Gaussian linear”);
- H_0 : Nonlinear transformation of an ARMA process;

(these are *composite* null hypotheses), look at statistics such as *time reversibility* $E[(X_n - X_{n-\tau})^3]$ or the non-linear prediction error (nlpe). The p -value can be computed with simulations. The bootstrap would estimate an ARMA model on the data and generate new data from the fitted model. **Constrained randomization** generates data with the same ARMA estimates, *i.e.*, with the same ACF. For instance, one can Gaussianize the data, compute its FFT, randomize the phase, transform back, and restore the distribution, but this tends to give a flatter power spectrum. Instead, start with a shuffled time series and iterate:

- Force the spectrum to be correct, by rescaling it to the correct amplitudes, keeping the phases;
- Force the distribution to be correct (resampling).

Conserving the Fourier amplitudes preserves the periodic ACF, not the ACF: to limit the problem, measure the corresponding artefacts and remove a few observations at the beginning or the end to minimize them. Alternatively, use simulated annealing to find a permutation of the original time series with a small discrepancy between the ACFs (not exactly the minimum discrepancy: that would be the original time series).

A large set of audio features for sound description (similarity and classification) in the Cuidado project **G. Peeters (2004)**

Audio features include:

- Spectral shape (linear or exponential fit; properties of the spectral density);
- ACF, zero crossing rate;
- Length of the ADSR (envelope) phases;
- Fundamental frequency, inharmonicity (weighted average distance between successive spectral peaks and multiples of the fundamental), harmonic deviation (deviation of the harmonics from the spectral envelope), odd/even harmonic ratio, tristimulus (relative energy in harmonics 1, 2–4 and 5+).

Time-varying market beta: does the estimation methodology matter? **B. Nieto et al. (2014)**

Comparison of several time-varying beta estimators (prefer Kalman or GARCH):

- Regression on a moving window, with constant or decreasing weights;
- Multivariate GARCH

$$\text{BEKK: } H_{t+1} = C'C + A'\varepsilon_t\varepsilon_t'A + B'H_tB$$

$$\text{DCC: } H_t = D_t R_t D_t$$

(where D_t is a diagonal GARCH model) and asymmetric variants;

- Kalman filters

$$\text{random walk: } \beta_{t+1} = \beta_t + \text{noise}$$

$$\text{random observations: } \beta_t = \beta_0 + \text{noise}$$

Arbitrated ensemble for time series forecasting **V. Cerqueira et al.**

To combine time series forecasts, model how each model performs for different types of inputs and how this performance changes with time (the metamodel forecasts the model error).

Graph-theoretic properties of the darkweb **V. Griffith et al.**

The Darkweb is only loosely connected: most sites have no outgoing links. Data from the tor2web proxy onion.link, crawled with scrapinghub.com (commercial), starting from directoryvi6plzm.onion and ahmia.fi/onions.

A divide-and-conquer framework for distributed graph clustering **W. Yang and H. Xu (2015)**

The generalized stochastic block model divides the nodes into clusters and outliers; the edge probability is p for nodes inside the same cluster and $q < p$ otherwise (edges between clusters or involving an outlier). It can be fitted, in parallel, with a divide-and-conquer approach:

- Randomly partition the nodes into groups;
- Find clusters in each of those subgraphs;
- Build a “fused” graph, whose nodes are the subgraph clusters, and with an edge between two clusters if the edge density exceeds some threshold t ($q < t < p$);
- Find clusters in the fused graph.

Real-time community detection in large social networks on a laptop **B. Chanberlain et al.**

The **minhash** of a subset $A \subset \llbracket 1, n \rrbracket$ is $h_\sigma(A) = \text{Min}\{\sigma(a), a \in A\}$ for a (fixed) random permutation $\sigma \in \mathfrak{S}_n$. The *Jaccard similarity* can be approximated with minhashes:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \mathbb{P}_{\sigma \in \mathfrak{S}_n} [h_\sigma(A) = h_\sigma(B)].$$

Given a large graph, the neighbours of a node can be compressed (lossily) using minhash signatures (e.g., 1000 hash functions for 1,000,000 nodes), and locality-sensitive hashing (LSH) can be used to find approximate nearest neighbours: with the Jaccard similarity,

they form a weighted graph, on which we can compute (local) communities. The Jaccard similarity also helps solve coverage problems (“find the smallest number of athletes that have influence over half of Twitter”) – $|A \cup B| = |A| + |B| - |A \cap B|$ links the number of neighbours of a community and of a node and their Jaccard similarity.

Querrying k -truss community in large and dynamic graphs
X. Huang et al. (2014)

The k -truss of a graph G is the largest subgraph in which each edge is in at least $k - 2$ triangles.

A k -truss community is a maximal k -truss subgraph in which any two edges are reachable through a series of adjacent triangles.

After putting the k -truss subgraphs in a suitable index, one can retrieve the k -truss community of a vertex in linear time.

Bayesian dynamic modeling and analysis of streaming network data
X. Chen and K. Irie (2015)

To model the flow of visitors on a graph (website), consider a non-Gaussian state space model

i : node

t : time

n_{it} : number of occupants of node i at time t

x_{ijt} : flow from i to j between $t - 1$ and t

$x_{0it} \sim \text{Pois} \phi_{it}$ new visitors

$x_{i,t} \sim \text{Mult}(n_{i,t-1}, \theta_{i,t})$ movement between nodes

ϕ_{it} : arrival rate

θ_{ijt} : transition probabilities

$$\theta_{ijt} = \frac{\phi_{ijt}}{\sum \phi_{i,t}}$$

$$\phi_{ijt} = \mu_t \alpha_{ij} \beta_{jt} \gamma_{ijt}$$

β_{jt} : attractivity of node j at time t

Unpaired image-to-image translation using cycle-consistent adversarial networks
J.Y. Zhu et al.

CycleGAN translates (unpaired) images between two domains (paintings and photographs, horses and zebras, summer and winter, aerial photographs and maps, etc.) by learning two mappings

$$\mathcal{X} \xrightleftharpoons[B]{G} \mathcal{Y}$$

with an adversarial loss to make the distribution of $G(\mathcal{X})$ indistinguishable from that of \mathcal{Y} , and to ensure $F \circ G \simeq \text{id}$ (cycle-consistency).

InfoGAN: interpretable representation learning by information maximizing generative adversarial nets
X. Chen et al.

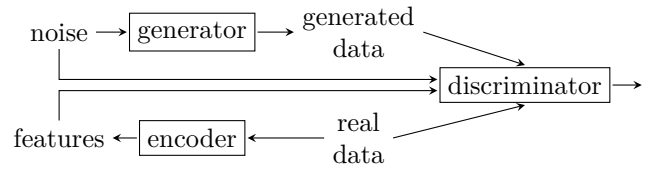
To make the latent space of GANs more interpretable (in an ICA-like way), split the noise into an incompressible part z and a “code” c , and solve the minimax problem

$$\begin{aligned} \min_G \max_D V_{\text{GAN}}(D, G) - \lambda I(c; G(z, c)) \\ V_{\text{GAN}}(D, G) = \mathbb{E}_{x \sim \text{Data}} [\log D(x)] + \\ \mathbb{E}_{(c, z) \sim \text{Noise}} [\log(1 - DG(c, z))] \end{aligned}$$

after replacing the mutual information with a (variational) lower bound.

Adversarial feature learning
J. Donahue et al. (2017)

The BiGAN has an encoder, converting the data to features, and its discriminator uses both the data and the latent representation.



Texture networks: feed-forward synthesis of textures and stylized images
D. Ulyanov et al (2016)

To generate a texture (target), find a CNN f generating similar VGG features, *i.e.*, minimizing

$$\| \text{VGG } f(\text{noise}) - \text{VGG target} \|.$$

For style transfer, use

$$\| \text{VGG } f(\text{noise}, \text{content}) - \text{VGG target} \|,$$

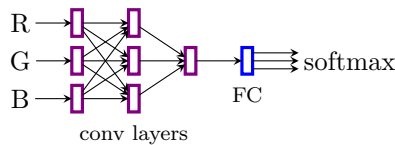
where the target is fixed.

Rethinking the inception architecture for computer vision
C. Szegedy et al.



X-CNN: cross-modal convolutional neural networks for sparse datasets
P. Veličković et al.

Process the input layers separately, mix them, and merge them.



Multiplicative LSTM for sequence modelling
B. Krause et al.

Tensor RNNs have character-specific hidden-to-hidden weights

$$h(t) = W_{hh}^{x_t} h_{t-1} + W_{hx} x_t.$$

Multiplicative RNNs use a sparse description of those weight matrices

$$W_{hh}^{x_t} = W_{hm} \cdot \text{diag}(W_{mx} x_t) \cdot W_{mh}.$$

This idea can be combined with LSTMs (which already contain a multiplication).

Google's multilingual neural machine translation system: enabling zero-shot translation
M. Johnson et al.

The same network can translate into many languages: just add a token, at the beginning of the sentence, specifying the target language.

Skip-thought vectors
R. Kiros et al.

Compute sentence embeddings by reconstructing the surrounding sentences, using a seq2seq RNN with GRU units, trained on the BookCorpus dataset.

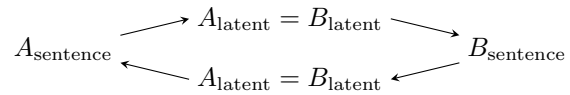
AudioSet: an ontology and human-labeled dataset for audio events
J.F. Gemmeke et al.

AudioSet is the audio equivalent of ImageNet: 600 classes, 2 million 10-second samples, from YouTube.

Unsupervised machine translation using monolingual corpora only
G. Lample et al.

Learn embedding of two languages into the same latent space, initialized with a bilingual lexicon (only words – no sentences), and learn to reconstruct a sentence from language A from a noisy version of its embedding. Add an adversarial regularization, with a discriminator to recognize the language from the latent space.

The translation loop is an example of noise.



Neural architecture search with reinforcement learning
B. Zoph and Q.V. Le (2017)

Reinforcement learning can help improve existing network architectures (number of layers, dimension, filter size, stride, etc.) and devise new LSTM-like cells.

Self-critical sequence training for image captioning
S.J. Rennie et al.

Reinforcement learning deals with non-differentiable rewards: it maximizes the *expected* reward wrt a baseline to reduce minibatch variance. Those ideas can be used for the non-differentiable evaluation metrics commonly used in natural language processing (BLEU, etc.) instead of cross-entropy.

Learning options in reinforcement learning
M. Stolle and D. Precup

Use frequently-visited states as subgoals for your reinforcement learning system to easily adapt to different goals.

Effect of reward function choices in risk-averse reinforcement learning
S. Ma and J.Y. Yu

Given a Markov decision process (MDP), one can maximize the value-at-risk (VaR) or the expected shortfall (ES, CVaR) instead of the expected discounted reward.

Hyperband: a novel bandit-based approach to hyperparameter optimization
L. Li et al. (2016)

To tune hyperparameters, *successive halving* uniformly allocates resources (number of training sets, number of features, iterations) to n random configurations, evaluates their performance, throws away the bottom half, and continues, improving the precision on the performance of the remaining configuration, until there is only one left. For a given time budget B , Hyperband tried different trade-offs between n and B/n .

Learning concept embedding with combined human-machine expertise
M.W. Wilber

Add triplet constraints $d_{ij} \leq d_{ik}$ to t-SNE (from MechanicalTurk).

***The high-dimensional geometry
of binary neural networks***
A.G. Anderson and C.P. Berg

In high dimensions, binarization approximately preserves directions and batch-normalized weight-activation dot products – but keep continuous weights for the first layer: it is very different.

***Exploring loss function topology
with cyclical learning rates***
L.N. Smith and N. Topin (2017)

Repeatedly increasing and decreasing the learning rate (e.g., $0.1 \rightarrow 1 \rightarrow 0.1$) allows the optimization to find several different minima, often better than a constant learning rate.

***Generating focused molecule libraries for drug
discovery with recurrent neural networks***
M.H.S. Segler et al.

Fit a recurrent neural net (3 stacked LSTM layers, 1024 dimensions, dropout=0.2) on the Smiles description of 1,000,000 molecules (a 1-dimensional, textual description of molecules) and use it to generate new molecules for further (in silico) testing.

***Kernel approximation methods
for speech recognition***
A. May et al.

Random features approximate the kernel trick. They can be selected iteratively: generate random features, fit the model, keep the best features; start again with a new set of random features.

The following loss functions do not penalize incorrect forecasts as much as cross-entropy:

$$\begin{aligned} & - \sum_{i,y} (\mathbf{1}_{y=y_i} + \beta p(y|x_i)) \log p(y|x_i) && \text{entropy-penalized} \\ & - \sum_i \log(p(y_i|x_i) + \lambda) && \text{capped} \\ & - \sum_{1 \leq i \leq k} \log p(y_i|x_i) && \text{top-}k. \end{aligned}$$

***World literature according to Wikipedia:
introduction to a DBpedia-based framework***
C. Hube et al. (2017)

DBpedia extracts data from Wikipedia infoboxes, using hand-written (crowd-sourced) rules. The data is not as clean as we would like: for instance, there is no single, reliable way of identifying “writers”.

***PathNet: evolution channels gradient descent
in super neural networks***
C. Fernando et al.

To learn different tasks on the same network (“evolutionary dropout”): pick a random set of paths through

the network, train the corresponding subnet, for the first task; freeze the corresponding weights; proceed to the second task.

***Deep forest: towards an alternative
to deep neural networks***
Z.H. Zhou and J. Feng (2017)

Deep models need not be limited to neural nets: use a random forest for a classification problem; concatenate its input and its output (class probabilities) and feed them to another random forest; iterate a few times. Those layers could be convolution-like, taking patches of an image or sequence as input.

***Batch renormalization:
towards reducing minibatch dependence
in batch-normalized models***
S. Ioffe

When using batch normalization with small or non-iid batches, add a (non-learned) affine transform to account for the difference in mean and variance between the minibatch and the population.

***PixelNet: representation of the pixels,
by the pixels, and for the pixels***
A. Bansal

For pixel-level predictions (semantic segmentation, edge detection, surface normal extraction), use a spatially-invariant (convolutional) model, with multi-scale features (“hypercolumns”, *i.e.*, connections from earlier, more detailed layers) and only sample a few (2000) pixels per image (to reduce the dependence, which could harm stochastic gradient descent).

***Cosine normalization: using cosine similarity
instead of dot product in neural networks***
L. Chunjie et al.

Replace the dot product $x \mapsto w \cdot x$ in neural networks with the cosine similarity (*i.e.*, correlation)

$$x \mapsto \frac{w \cdot x}{\|w\| \|x\|}.$$

***Neural audio synthesis of musical nodes
with WaveNet autoencoders***
J. Engel et al.

The NSynth dataset contains 300,000 4-second notes, from 1000 instruments, from commercial sample libraries, with different pitches and velocities.

***Learning character-level compositionality
with visual features***
F. Liu et al.

For alphabetical languages, character-level models can understand word morphology and deal with unknown words. For more complex writing systems (Chinese,

Japanese, Korean), convert the characters to 36×36 images and feed them to a CNN + GRU-CNN.

Tacotron: towards end-to-end speech synthesis
Y. Wang et al.

End-to-end speech synthesis by combining convolution (n -grams), batch normalization, residual connections, highway networks, bidirectional GRUs and attention to produce spectrogram frames.

A study of complex deep learning networks on high-performance, neuromorphic and quantum computers
T.E. Potok et al.

Adiabatic quantum computers can fit (non-restricted, *i.e.*, non-bipartite) Boltzmann machines – networks with intra-layer connections are becoming tractable.

Neuromorphic computers (spiking neural nets, SNN), built with memristors, may make computations more efficient.

Opening the black box of deep neural networks via information
R. Schwartz-Ziv and N. Tishby

Representing a learning (deep) neural net in the *information plane* ($I(\text{input}; \text{hidden})$, $I(\text{hidden}; \text{output})$) where the mutual information $I(X, Y) = \text{KL}(p_{x,y} \| p_x p_y)$ measures the information the input variable X has on the output Y , reveals two phases:

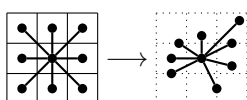
- Compressing the information, *i.e.*, increasing $I(\text{hidden}; \text{output})$ while preserving $I(\text{input}; \text{hidden})$;
- Discarding irrelevant information, *i.e.*, decreasing $I(\text{input}; \text{hidden})$, which helps generalization.

Database learning: towards a database that becomes smarter every time
Y. Park et al.

The answer to each database query can help fine-tune a probabilistic model of the data, used to speed up future queries and/or provide faster approximate results.

Active convolution: learning the shape of convolution for image classification
Y. Jeon and J. Kim

Active convolution units have no fixed shape and allow for fractional pixel coordinates.



A simple neural network module for relational learning
A. Santoro et al.

Relation networks (RN) are built from modules of the form

$$x \mapsto f_{\phi} \left(\sum_{ij} g_{\theta}(x_i, x_j) \right)$$

where f and g are multi-layer perceptrons (but it is the same g_{θ} for all pairs).

Fast algorithms for convolutional neural networks
A. Lavin and S. Gray

The *Winograd minimal filtering algorithm* computes 1- or 2-dimensional filters with the minimal number of multiplications – for the small filters used on CNNs, it is faster than the FFT.

HyperNetworks
D. Ha et al.

Hypernetworks are networks generating the weights of another network; they can be trained end-to-end with backpropagation. They can generate non-shared LSTM (or CNN) weights.

Deep feature interpolation for image content changes
P. Upchurch et al.

Deep neural networks can describe images with features in a linear (not curved) space: linear interpolation in a direction (age, gender, glasses, smile, etc.) produces high-resolution image transforms.

Fitted learning: models with awareness of their limits
N. Kardan and K.O. Stanley

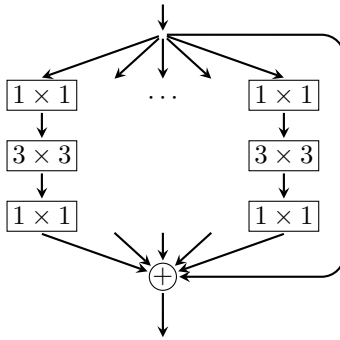
For classification problems, use k neurons for each class in the output layers (instead of 1), train to set their values to 1, but use their *product* at test time – this limits overly-confident generalizations.

Unsupervised learning for physical interaction through video prediction
C. Finn et al.

Predict motion, at the pixel level, assuming that pixels do not move large distances, and that pixels are only transformed in a small number of ways (pixels belonging to the same object move in the same way).

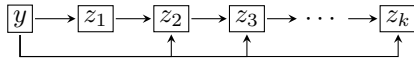
Residual networks behave like ensembles of relatively shallow networks
A. Veit et al.

**Aggregated residual transformations
for deep neural networks**
S. Xie



Input convex neural networks
B. Amos et al.

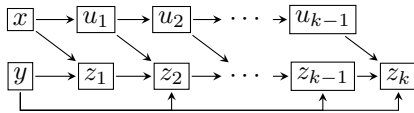
Neural nets can be constrained to be convex functions of their input, by requiring the weights to be positive



(skip connections are useful for the network to remain expressive), or of part of the input,

$$z_{i+1} \sim z_i \odot u_i + z_i \odot y + u_i$$

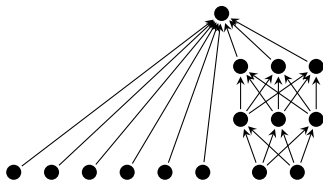
by requiring part of the input (the $z \odot u$ term) to be positive.



Such networks can be used in optimization problems,

$$\underset{y}{\text{Argmin}} f(x, y; \theta).$$

**Wide and deep learning
for recommender systems**
H.T. Cheng et al.



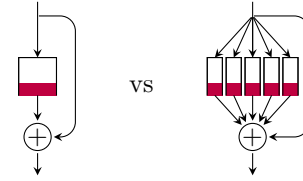
Self-normalizing neural networks
G. Klambauer

No batch normalization is needed for the activation function

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha(e^x - 1) & \text{if } x \leq 0. \end{cases}$$

**Aggregated residual transformations
for deep neural networks**
S. Xie et al.

Put the non-linearity (purple) before or after aggregating.



Memory networks
J. Weston et al. (2015)

Add (classical) memory cells to a neural network: at each time step, the neural network decides which cells to update and how, using their contents together with the new data.

DeepCoder: learning to write programs
M. Balog et al.

Inductive program synthesis (IPS) for a simple array manipulation language (only linear control flow: head, last, take, drop, access, min, max, rev, sort, sum; map, filter, count, zip, scan; $(+1)$, (-1) , $(\times 2)$, $(/2)$, $(\times -1)$, $(^2)$, (> 0) , (< 0) , odd, even). The neural network is only used to guide more traditional approaches: stochastic local search, depth-first search, etc.

mixup: beyond empirical risk minimization
H. Zhang et al.

To help the network generalize better to examples slightly outside the training distribution, augment the data with convex combinations $(1 - \lambda)(x_1, y_1) + \lambda(x_2, y_2)$ (use one-hot-encoding for y).

Coherent line drawing
H. Kang et al.

The *edge tangent flow* of an image I is computed iteratively:

$$t_x \leftarrow \nabla I(x)^\perp$$

$$t_x \leftarrow \frac{1}{k} \sum_{\|x-y\| \leq r} \sigma(\|\nabla I(x)\| - \|\nabla I(y)\|)(t_x \cdot t_y) t_y.$$

Applying a difference of Gaussians (DoG) in the direction orthogonal to the edge flow gives a nice line drawing.

Simplex noise demystified
S. Gustavson (2005)

Simplex noise is similar to Perlin noise, but replaces hypercubes with simplices, which facilitates interpolation.

SATGraf: visualizing the evolution of SAT formula structures in solvers
Z. Newsham et al. (2014)

To visualize how SAT solvers transform a boolean formula, look at the *clause-variable incidence graph* – its community structure is related to the running time.

Network risk and financial crisis
Y.A. Chen (2015)

Build a parametric portfolio from the Katz centrality

$$\mathbf{x} = \gamma \mathbf{A} \mathbf{x} + \beta \mathbf{1}$$

of the weighted network defined by the variance of residual returns.

Compute the beta of each stock wrt that portfolio; decompose the idiosyncratic volatility into “network volatility” and a remainder.

Media network and return predictability
L. Guo and Y. Tao (2017)

The connection score of two stocks i, j , is

$$CS_{ij} = \sum_k \text{tone}_{ik} \times \text{tone}_{jk}$$

where the sum is over all news mentioning both i and j and the tone is in $[-1, 1]$.

The media connection index

$$\frac{\sum_{i \neq j} CS_{ij}}{\sum_{ij} CS_{ij}}$$

can be used as a risk measure.

Deep stock representation learning: from candlestick charts to investment decisions
G. Hu et al.

Feed candlestick charts (*i.e.*, images), for the FTSE 100 stocks, to a convolutional autoencoder; cluster them using the resulting latent representation, with a community detection algorithm (no need to specify the number of clusters); invest in the stocks with the largest Sharpe ratio in the cluster.

Local explosion modelling by noncausal processes
C. Gouriéroux and J.M. Zakoian (2016)

Non-causal AR(1) processes can model bubbles growing and bursting; aggregating them allows for different rates of increase (*e.g.*, a continuous distribution); one can also add a (causal, Gaussian) AR(1).

The myths and limits of passive hedge fund replication
N. Amenc et al. (2007)

There are two approaches to replicate hedge funds (neither works well):

- Compute the exposure of hedge fund returns to a set of pre-selected factors;
- Build an option, on a reference asset (anything, *e.g.*, S&P 500) and choose the payoff function giving the desired payoff distribution (“moment matching”). The price of that option can be used as a measure of performance. This “replicating” strategy has comparable returns, but not at the same times. The first moment is not captured: the returns are lower. [There is a bivariate, copula-based variant but it only models (portfolio, underlying), not (portfolio, fund).]

Possible improvements to capture nonlinearities, dynamic trading, non-stationarity and time series properties include:

- Heuristic dynamic (rule-based) strategies;
- Instrumental variables for time- and state-dependence;
- Future factor loading forecasts;
- Time-varying models (Kalman filter, particle filter, Markov switching);
- Ad hoc options portfolios (*e.g.*, ATM and OTM puts and calls on S&P 500);
- Statistical techniques to estimate the options (and underlyings) needed.

An alternative approach to alternative beta
T. Toncalli and J. Teïletche (2007)

Use the Kalman filter to estimate time-dependent factor exposures. It also decomposes the returns (for one period) into traditional (*i.e.*, average) beta, alternative (difference with the average) beta and alpha (residual, intercept).

Tracking problems, hedge fund replication and alternative beta
T. Roncalli and G. Weisang (2008)

The Kalman filter provides (Gaussian, linear) factor exposures. The unscented Kalman filter (UKF) allows for non-Gaussian innovations. One can add non-linear assets to the list of factors; if they depend on a parameter (say, a strike), it can be modeled as a random walk, as part of the hidden state, but the model becomes nonlinear – use a particle filter (PF, sequential Monte Carlo, SMC). Hedge fund performance can be decomposed into: traditional beta, alternative beta, lag (alternative beta one period ahead), alternative alpha (the rest – this includes illiquid assets).

Regular(ized) hedge fund clones
D. Giamouridis and S. Paterlini (2009)

Use the lasso for factor-based hedge fund replication [the resulting weights are biased towards zero].

***Can hedge fund returns be replicated?
The linear case***
J. Hasanhodzic and W. Lo

Replicate individual hedge funds using in-sample or 24-month rolling window regressions against USD, AA bonds, BAA—treasury, S&P 500, commodities (GSCI) and VIX monthly changes, with no intercept, weights summing to 1, and using leverage to match the target’s volatility. The clones have lower autocorrelation (a proxy for illiquidity).

***Hedge fund replication:
putting the pieces together***
V. Weber and F. Peres (2013)

Do not only use static factors, but also dynamic ones (carry, momentum). Correct hedge fund returns for illiquidity:

$$\hat{R}_t = \frac{R_t - \alpha R_{t-1}}{1 - \alpha} \quad (\text{Gettner})$$

R_t = observed returns
 α = $\text{Cor}(R_t, R_{t-1})$.

***Send in the clones?
Hedge fund replication using futures contracts***
N.P.B. Bollen and G.S. Fisher (2012)

Replicating hedge fund indices [many people invest in funds of funds] with futures (USD, 10-year, gold, oil, S&P 500) using 1 to 4 year rolling window regressions gives poor performance and shows no sign of market timing, $r \sim \sum f_i + \sum 1_{f_i > 0} f_i$; the clone-target correlation is high, but mostly due to market exposure.

***Hedge fund replication:
a model combination approach***
M.S. O’Doherty et al. (2016)

Instead of directly estimating a linear replication model, estimate separate replication models, using domestic equities (S&P 500, S&P Midcap 400), international equities (FTSEE 100, Nikkei 225), currencies (GBP, CHF, JPY) and commodities (gold, corn, oil) and average using “log-scale” weights (BMA).

A primer on alternative risk premia
R. Hamdan et al. (2016)

Five concerns with the five-factor model
D. Blitz et al.

The two factors added to the Fama-French model (market, size, value, profitability, investment) are not robust, lack an economic rationale, and still ignore momentum.

***Bayesian Poisson Tucker decomposition
for learning the structure
of international relations***
A. Schein et al. (2016)

Model international relations (country i takes action a on country j at time t) using country-community, action-topic and time-regime factor matrices:

$$y_{i \rightarrow j}^t \sim \text{Poisson} \left(\sum_c \theta_{ic} \sum_d \theta_{jd} \sum_k \phi_{ak} \sum_r \psi_{tr} \lambda_{c \rightarrow d}^r \right)$$

with Gamma priors, estimated with MCMC.

A gentle introduction to value at risk
L. Ballotta and G. Fusai (2017)

The idiosyncratic momentum anomaly
D. Blitz et al. (2017)

Use idiosyncratic momentum (rather than total momentum) to forecast future returns.

***Deep learning for forecasting stock returns
in the cross-section***
M. Abe and H. Nakayama (2018)

Deep neural networks, with 1 to 6 fully-connected hidden layers, 50% dropout, tanh activations, to forecast (uniformized) 1-month ahead stock returns for MSCI Japan, from 25 (uniformized) investment factors, both current and past values (3, 6, 9 and 12 months), trained for 100 epochs on universe-sized minibatches, on a 10-year window.

***Proofs are programs:
19th century logic and 21st century computing***
P. Wadler (2000)

Frege’s logic considers judgments (e.g., $\vdash A$), axioms (e.g., $\vdash A \rightarrow A$) and deduction rules, e.g.,

$$\frac{\vdash B \rightarrow A \quad \vdash B}{\vdash A}.$$

Gentzen’s logic adds assumptions to the judgments (e.g., $B_1, \dots, B_n \vdash A$ instead of $\vdash A$) and to the deduction rules, e.g.,

$$\frac{\Gamma \vdash B \rightarrow A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A}.$$

While this may look more complex (in particular, the three if-then notions: \rightarrow for propositions, \vdash for judgments and \div in the inference rules), the deduction rules end up simpler and more intuitive.

Typed λ -calculus has typing judgments (e.g.,

$$x_1 : B_1, \dots, x_n : B_n \vdash t : A,$$

meaning that if the x_i ’s have types B_i , then t has type A) and the same deduction rules as Gentzen’s system, e.g.,

$$\frac{\Gamma \vdash t : B \rightarrow A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash t(u) : A}.$$

(Unless we explicitly add recursion, all functions terminate and the calculus is not Turing-complete.) Term reduction corresponds to proof simplification (Curry-Howard correspondance).

***Physics, topology, logic and computation:
a Rosetta stone***

J.C. Baez and M.M. Stay (2009)

Variants of monoidal categories

- braided: $\swarrow \searrow$
- symmetric: $\swarrow \searrow = \searrow \swarrow$
- cartesian: $\otimes = \times$
- closed: $\otimes \dashv \underline{\text{Hom}}$
- compact: $\underline{\text{Hom}}(X, Y) = X^* \otimes Y$
- dagger (with a functor $\dagger : \mathfrak{C} \rightarrow \mathfrak{C}^{\text{op}}$, the identity on $\text{Ob } \mathfrak{C}$, with $f^{\dagger\dagger} = f$)

appear in many domains, e.g.,

- cobordisms
- tangles in \mathbf{R}^n
- finite-dimensional Hilbert spaces (the inner product gives a dagger structure)
- finite-dimensional representations of a Lie (resp. quantum) group form a compact symmetric (braided) monoidal category.

Propositional calculus forms a monoidal category with propositions as objects, a single morphism $X \rightarrow Y$ whenever $X \Rightarrow Y$, and $\otimes = \wedge$, $\underline{\text{Hom}}(X, Y) = X \Rightarrow Y$. It is a **Heyting algebra**, *i.e.*, a preorder \mathfrak{C} such that \mathfrak{C} and \mathfrak{C}^{op} be cartesian closed.

Linear logic replaces $\wedge, \Rightarrow, \top$ with \otimes, \multimap and I and only requires a symmetric monoidal category, not a cartesian closed one: there are no natural morphisms $X \rightarrow X \otimes X$ or $X \rightarrow I$, *i.e.*, we cannot duplicate or delete information (as in quantum physics or chemistry).

Alternatively, one can consider propositions as objects and proofs as morphisms, where the proofs are obtained from inference rules, *i.e.*, natural or *dinatural* transformations, most of which come from the monoidal/braided/cartesian structure.

In **λ -calculus**, everything is a function. For instance, **Church numerals** are

$$\begin{aligned} 0 &= \lambda f \mapsto \lambda x \mapsto x & f &\mapsto \text{id} \\ 1 &= \lambda f \mapsto \lambda x \mapsto f(x) & f &\mapsto f \\ 2 &= \lambda f \mapsto \lambda x \mapsto f(f(x)) & f &\mapsto f^2 \end{aligned}$$

and addition and multiplication are

$$\begin{aligned} \text{times} &= \lambda a \lambda b \lambda f \lambda x \ a(b(f)) \\ \text{plus} &= \lambda a \lambda b \lambda f \lambda x \ a(f)(b(f)(x)). \end{aligned}$$

(I write $\lambda x \mapsto t$ or $\lambda x \ t$ instead of $(\lambda x.t)$ and dispense with many of the parentheses.)

Typed λ -calculus, where every term has a type, forms a cartesian closed category with types as objects and equivalence classes of terms as morphisms (if we

do not focus on *what* is computed but *how* it is computed, we get a 2-category, with terms as morphisms and equivalence classes to rewrites as 2-morphisms).

Linear type theories correspond to symmetric monoidal categories. They can also be described by **combinators**. The usual ones

$$\begin{aligned} I &= \lambda x \mapsto x \\ K &= \lambda x \mapsto \lambda y \mapsto x \\ S &= \lambda x \mapsto \lambda y \mapsto \lambda z \mapsto x(z)(y(z)) \end{aligned}$$

cannot do because K forgets information and S duplicates it. Each term t is equivalent to $\text{cp}(t) \text{vp}(t)$, where $\text{cp}(t)$ is the combinator part and $\text{vp}(t)$ the variable part.

Internal set theory
E. Nelson (2002)

There are two ways of presenting non-standard analysis: either explicitly, by constructing the hyperreals, or by tweaking the ZFC axioms (*internal set theory*):

- The subset axiom, to define $\{x \in X : A(x)\}$, is only valid for internal formulas A (those which do not use the “standard” predicate);
- Transfer: $\forall^{\text{st}} t_1 \dots \forall^{\text{st}} t_n (\forall^{\text{st}} x A \Leftrightarrow \forall x A)$, for any internal formula A ;
- Idealization: $\forall^{\text{stfin}} X \exists y \forall x \in X A \Leftrightarrow \exists y \forall^{\text{st}} x \in X A$ (for A internal);
- Standardization:

$$\forall^{\text{st}} X \exists^{\text{st}} Y \forall^{\text{st}} z (z \in Y \Leftrightarrow z \in X \wedge A(z))$$

for any formula A (internal or external), which allows us to define $^S\{z \in X : A(z)\}$ the standard set whose standard elements are the standard elements z of X such that $A(z)$.

The “standard” predicate is no longer limited to real numbers, but applies to sets, functions, topological spaces, points in topological spaces, etc.

Includes exercises. Check *Radically elementary probability theory*, by the same author, on probability theory with non-standard finite probability spaces.

Division by three
P.G. Doyle and J.H. Conway (1994)

Proving $A \times 3 \approx B \times 3 \implies A \approx B$ does not require the axiom of choice.

***Developing bug-free machine learning systems
with formal mathematics***
S. Selsam et al. (2017)

The **lean** theorem prover tries to combine program verification (à la Coq) and formal mathematics (à la Mizar). Here, it is used for a provably correct gradient descent implementation.

**The ring of algebraic functions
on persistence bar codes
A. Adcock et al. (2013)**

To turn a barcode $[x_1, y_1], \dots, [x_n, y_n]$ into features, for machine learning, use multisymmetric polynomials $p_{ab} = \sum_i x_i^a y_i^b$, for instance

$$\begin{aligned} & \sum x_i(y_i - x_i) \\ & \sum (y_{\max} - y_i)(y_i - x_i) \\ & \sum x_i^2(y_i - x_i)^4 \\ & \sum (y_{\max} - y_i)^2(y_i - x_i)^4. \end{aligned}$$

Barcodes form an affine (semi)algebraic ind-variety, with ring of algebraic functions $\Lambda_2 = \varinjlim k[x_1, y_1, \dots, x_n, y_n]^{\mathfrak{S}_n}$ (multisymmetric polynomials), quotiented by an ideal $D = \varinjlim D_n$ corresponding to the identification of intervals of length zero. Λ_2 is freely generated by the multisymmetric power sums $p_{ab} = \sum_i x_i^a y_i^b$, and D by the $p_{a+1,b} - p_{a,b+1}$.

An alternative is to use the bottleneck or Wasserstein distance.

**Persistent homology
H. Edelsbrunner and D. Morozov
(Handbook of discrete
and computational geometry, 2017)**

The implementation details are not that complicated.

**High-dimensional topological data analysis
F. Chazal (Handbook of discrete
and computational geometry, 2017)**

1. A subset $X \subset \mathbf{R}^n$ can be represented by its distance function

$$\phi : \begin{cases} \mathbf{R}^n & \rightarrow \mathbf{R}_+ \\ x & \mapsto d(x, X) \end{cases}$$

(it is proper and $x \mapsto \|x\|^2 - \phi(x)^2$ is convex). To increase the robustness to outliers, consider measures instead of subsets, e.g., $\mu = \sum_{x \in X} \delta_x$ if X is discrete, and the functions

$$\delta_{\mu, \ell} : \begin{cases} \mathbf{R}^n & \rightarrow \mathbf{R}_+ \\ x & \mapsto \inf\{r > 0 : \mu(B(x, r)) > \ell\} \end{cases}$$

They are not smooth, but the **distance to measure** $d_{\mu m}^2$ is

$$d_{\mu m}^2 : \begin{cases} \mathbf{R}^n & \rightarrow \mathbf{R}_+ \\ x & \mapsto \frac{1}{m} \int_0^m \delta_{\mu, \ell}(x)^2 d\ell \end{cases}$$

(it is the average of the squared distances from x to its k nearest neighbours). If two measures μ and ν are close (for the Wasserstein distance), then the sublevel sets of their distance-to-measure functions are homotopy equivalent (**reconstruction** theorem).

2. For many tasks, full reconstruction is not needed and a few homotopy or **homology** invariants suffice.

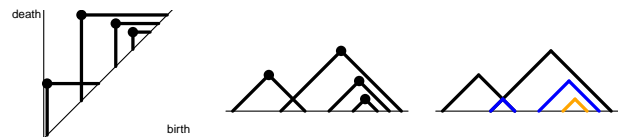
In particular, the **nerve** of an open cover $X = \bigcup_i U_i$ all of whose finite intersections are contractible is homotopy equivalent to X (nerve theorem).

3. **Persistent homology** keeps track, not only of the dimensions of the homology groups (Betti numbers) of $\bigcup_{x \in X} B(x, \varepsilon)$ as ε increases, but also of individual connected components, cycles and cavities – the result, the $(\text{birth}_i, \text{death}_i)$ pairs, forms the **persistence diagram**. The **bottleneck distance** between two persistence diagrams D and D' is the smallest $\delta > 0$ for which we can match all points of D at distance at least δ from the diagonal with a point of D' at distance at most δ . It is bounded by twice the Gromov-Hausdorff distance.

4. The **Mapper** algorithm visualizes a dataset $X \subset \mathbf{R}^n$ and a function $f : X \rightarrow \mathbf{R}$ by choosing an open cover $\bigcup I_i$ of $f(X) \subset \mathbf{R}$ by intervals, clustering the points in each $f^{-1}(I_i)$ into a (finite) cover, and plotting the 1-skeleton of the corresponding cover of X .

**Topological analysis of financial time series:
landscapes of crashes
M. Gidea and Y. Katz (2017)**

The **persistence landscape** is obtained from the (birth, death) persistence diagram by flipping it 45°, associating a piecewise function to each birth-death pair, and considering the sequence of functions given by the k th largest values. While persistence diagrams form a (non-complete) metric space for the Wasserstein distance, persistence landscapes form a Banach space for the L^p norm. (Measure theory on an infinite dimensional Banach space is a little more complicated: for instance, the notions of weak (Pettis) and strong (Bochner) integrability differ.)



The L^1 or L^2 norm of the persistence landscape of the daily returns of a few indices (S&P 500, DJIA, Nasdaq, Russel 2000) on a 50-day window may help forecast crises.

**Statistical topological data analysis
using persistence landscapes
P. Bubenik (2015)**

Initial paper on persistence landscapes.

**A persistence landscapes toolbox
for topological statistics
P. Bubenik and P. Dłotka**

Standalone, file-based C++ implementation (computing persistence landscapes, their averages, their L^p distances). Also check the TDA R package.

Topology-based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival
M. Nicolau et al. (2011)

The **mapper** algorithm takes a cloud of points \mathcal{X} and a (Morse) function $f : \mathcal{X} \rightarrow \mathbf{R}$ (e.g., a random projection, a measure of density, a distance to a baseline, an eccentricity measure), applies a clustering algorithm to the points in each $f_a = f^{-1}([a - \varepsilon, a + \varepsilon])$, and builds a graph with those clusters as nodes and edges between $x \in f_a$ and $y \in f_b$ if $x \cap y \neq \emptyset$ (partial clustering, discrete Morse theory). In R, check **TDAmapper**.

It is another way of “filtering” a cloud of points into a graph, keeping most of its features (loops, appendages, etc.).

Topological methods for the analysis of high dimensional datasets and 3D object recognition
G. Singh et al.

In the Mapper algorithm, using several Morse functions defines a simplicial complex instead of a graph – and letting ε vary leads to persistence homology.

The Topology Toolkit
J. Tierny et al.

Topological data analysis (TDA) is not a single algorithm (Mapper, or persistence homology, depending in who you ask), but a smogasbord of techniques. TTK is a library (C++, BSD, accessible from Python) and graphical toolkit (built on Paraview) using low-dimensional TDA (mostly discrete Morse theory) to visualize scientific data (PDE solutions and other scalar fields).

The data is given as a Morse function $f : \mathcal{M} \rightarrow \mathbf{R}$, i.e., a scalar field (or several of them); in contrast with high-dimensional TDA, the PL manifold \mathcal{M} is often uninteresting (\mathbf{R}^2 or \mathbf{R}^3).

TDA studies the transitions of the Betti numbers $\beta_k f^{-1}\{x\}$ or $\beta_k f^{-1}]-\infty, x]$ or $\beta_k f^{-1}[x, +\infty[$ as x varies.

A point is *critical* if one of

$$\begin{aligned} \text{Link}^+(p) &= \{q \in \text{Neigh}(p) : f(q) > f(p)\} \\ \text{Link}^-(p) &= \{q \in \text{Neigh}(p) : f(q) < f(p)\} \end{aligned}$$

is not simply connected. Critical points can be paired into birth-death pairs, defining a *persistence diagram*. To simplify f , one can prune short-lived pairs (those whose persistence, $f(\text{death}) - f(\text{birth})$ is small). The *persistence curve* is the plot of the number of critical points with persistence at least x , as x increases.

The *Reeb graph* segments \mathcal{M} into regions where $\pi_0 f^{-1}c$ does not change.

The *Morse complex* is defined by the attraction bassins of f .

Subsampling methods for persistent homology
F. Chazal et al. (2015)

Compute the persistence landscape on several subsamples and average them.

Equilibrated adaptive learning rates for non-convex optimization
Y.N. Dauphin et al.

In the presence of different curvatures, in particular around a saddle point, gradient descent oscillates a lot. **Preconditioning** uses a change of variables,

$$\begin{aligned} \theta &= D^{-1/2} \hat{\theta} \\ \hat{f}(\hat{\theta}) &= f(D^{-1/2} \hat{\theta}) = f(\theta) \\ \nabla \hat{f} &= D^{-1/2} \nabla \\ \nabla^2 \hat{f} &= D^{-1/2'} (\nabla^2 f) D^{-1/2} \\ \theta_{t+1} &\leftarrow \theta_t - \eta D^{-1} \nabla f(\theta_t). \end{aligned}$$

The *absolute Hessian* (defined from the eigen-decomposition of the Hessian, by taking the absolute value of the eigenvalues) gives a perfect conditioner (the curvature is the same in all directions), $D = |H|$, but it is computationally expensive.

The *Jacobi conditioner* is the diagonal of the Hessian. It could be estimated as

$$\text{diag } H = \mathbb{E}_{v \sim \text{Unif}(\{\pm 1\}^n)} [v \odot H v].$$

AdaGrad, AdaDelta, RMSProp are other diagonal conditioners.

The *equilibrium conditioner* is the diagonal of the absolute Hessian; it can be defined as the L^2 norm of the rows of the Hessian; it can be estimated as

$$\|H_{i\cdot}\|^2 = \mathbb{E}_{v_i \sim N(0,1)} [(Hv)^2].$$

A smart stochastic algorithm for nonconvex optimization with applications to robust machine learning
A. Aravkin and D. Davis

Machine learning minimizes $\sum_{1 \leq i \leq n} f_i(x)$. Trimmed machine learning minimizes $\sum_{1 \leq i \leq k} f_{(i)}(x)$, where $f_{(i)}(x)$ are the order statistics of $(f_i(x))_i$.

The problem can be formulated as

$$\begin{aligned} \text{Find} \quad & w \in \mathbf{R}^n, x \in \mathcal{H} \\ \text{To minimize} \quad & \frac{1}{n} \sum_i w_i f_i(x) + r_1(w) + r_2(x) \\ \text{Where} \quad & r_1 = \mathbf{1}_{\Delta_h} \\ & \Delta_h = \{w \in [0, 1]^n : w' \mathbf{1} \leq h\}. \end{aligned}$$

and solved by proximal alternating minimization ($w \leftarrow \text{prox}_{r_1}(\dots)$, $x \leftarrow \text{prox}_{r_2}(\dots)$). With mini-batches, use variance reduction (SVRG).

Model calibration with neural networks
A. Hernandez (2016)

Given enough training data, optimization can be replaced with a trained neural net.

Pymanopt: a Python toolbox for optimization on manifolds using automatic differentiation
J. Townsend et al. (2016)

Automatic differentiation can also compute (first and second) derivatives on Riemannian manifolds.

Learning to learn for global optimization of black box functions
Y. Chet et al.

A recurrent neural network (RNN: LSTM or differentiable computer) can learn to reproduce what a Bayesian blackbox optimizer does (it uses its memory to store information about the previous function evaluations). It is much faster than GP-based algorithms, and competitive in case of model mis-specification.

Understanding deep learning requires rethinking generalization
C. Zhang et al.

We still do not know why deep neural networks work well: VC dimension, Rademacher complexity, uniform stability, explicit/implicit regularization, model capacity fail to explain why they generalize so well in some cases (real data) but not others.

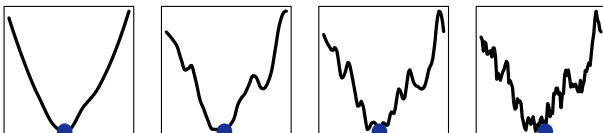
A tutorial on Bayesian optimization of expensive cost functions, with applications to active user modeling and hierarchical reinforcement learning
E. Brochu et al. (2010)

An entropy search portfolio for Bayesian optimization
B. Shahriari et al.

Use a portfolio of acquisition functions and follow that giving the largest decrease in the uncertainty of the location of the minimizer (or sample accordingly).

A theoretical analysis of optimization by Gaussian continuation
H. Mobahi and J.W. Fisher (2015)

Optimization by continuation [or mollifying] starts to optimize an easy simplification (e.g., a convex relaxation) of the problem, and progressively transforms it into the actual task.



The irace package: iterated racing for automatic algorithm configuration
M. López-Ibáñez et al.

Implementation of the iterated F-race algorithm, with a few extensions (other statistical tests, restarts, etc.).

Guaranteed non-convex optimization: submodular maximization over continuous domains
A.A. Bian et al. (2016)

The notion of *submodularity* can be generalized to continuous domains: for $f : \mathbf{R}^n \rightarrow \mathbf{R}$, the following conditions are equivalent.

- (i) $f(x) + f(y) \geq f(x \wedge y) + f(x \vee y)$, where \wedge and \vee are the coordinate-wise min and max;
- (ii) $\forall x \forall i \neq j \frac{\partial^2 f}{\partial x_i \partial x_j} \leq 0$;
- (iii) $f(\lambda e_i + a) - f(a) \geq f(\lambda e_i + b) - f(b)$, whenever $a \leq b$, $a_i = b_i$, $\lambda \geq 0$, where e_i is the i th basis vector.

Best subset selection via a modern optimization lens
D. Bertsimas et al. (2015)

Best subset selection for linear regression can be done exactly for $p \leq 30$ (**leaps**), with the lasso, or with *non-convex penalized regression*; the minimax concave penalty (MCP), implemented in **Sparsenet**, is quadratic for β small and then constant; the smoothly clipped absolute deviation (SCAD) penalty is linear, then quadratic, then constant.

Progress in mixed integer programming makes solving the exact problem reasonable for $p = 100$, $n = 1000$.

Hard thresholding computes $H_k(x) = \underset{\|\beta\|_0 \leq k}{\text{Argmin}} \|\beta - c\|_2^2$.

If g is Lipschitz and convex,

$$g(\eta) \leq g(\beta) + \frac{L}{2} \|\eta - \beta\|_2^2 + \langle \nabla g(\beta), \eta - \beta \rangle.$$

To solve

$$\underset{\beta}{\text{Minimize}} g(\beta) \text{ subject to } \|\beta\|_0 \leq k$$

for g lipschitz and convex, iterate

$$\beta_{m+1} \leftarrow H_k\left(\beta_m - \frac{1}{L} \nabla g(\beta_m)\right)$$

(discrete projected gradient) until $\|\Delta\beta\|_2 \leq \varepsilon$; then solve the continuous problem for the corresponding non-zero coordinates.

A line search $\beta_{m+1} = (1 - \lambda)\beta_m + \lambda H$ gives better empirical performance.

Subgroup discovery
M. Atzmueller (2005)

Subgroup discovery (SD) is a frequent-itemset-based supervised clustering technique: after extracting frequent association rules for the target variable, select a small number of them with good quality (measured by some statistical test: χ^2 , gof, etc.) so that they hardly overlap and have good coverage.

***Anytime discovery of a diverse set of patterns
with Monte Carlo Tree search***
G. Bosc et al.

Pattern mining is a form of structured learning (progressively build patterns, one element at a time): Monte Carlo Tree Search (MCTS) produces more diverse rules than greedy approaches.

***On b-bit min-wise hashing for large-scale
regression and classification with sparse data***
R.D. Shah and N. Meinshausen (2016)

Large-scale datasets (large p , large n , but sparse) can be tackled with random projections ($X \mapsto XA$, A random) or sketches ($X \mapsto AX$). *b-bit min-wise hashing* reduces a large sparse binary matrix X as follows:

- Pick a random permutation of the columns;
- In each row, find the index (in the permuted matrix) of the first non-zero value;
- Use the last b bits of those indices as b new columns in the compressed matrix;
- Repeat a few times with more permutations.

“Similarity” between rows is preserved.

***Sparse solutions to nonnegative linear systems
and applications***
A. Bhaskara et al.

To solve $Ax = b$, where A , x , b have nonnegative coefficients and $\|b\|_1 = 1$, increment x coordinatewise while keeping the “potential” $\Phi(x) = \sum_j b_j(1 + \delta)^{(Ax)_j/b_j}$ small; a small number of iterations leads to a sparse approximate solution.

***Phase recovery from a Bayesian point of view:
the variational approach***
A. Drémeau and F. Krzakala

The phase recovery problem, finding $x \in \mathbb{C}^n$ such that $y = |Dx|$, can be solved by alternating projections, convex relaxation, or variational Bayes.

***Katyusha: the first direct acceleration
of stochastic gradient methods***
Z. Allen-Zhu (2016)

Katyusha momentum combines variance reduction and

momentum.

$$\begin{aligned}x_{n+1} &\leftarrow \lambda z_n + \mu x^* + (1 - \lambda - \mu)y_n \\i &\sim U(\llbracket 1, n \rrbracket) \\ \nabla_{n+1} &\leftarrow \nabla f(x^*) + \nabla f_i(x_{n+1}) - \nabla f_i(x^*) \\ y_{n+1} &\leftarrow x_{n+1} - \beta \nabla_{n+1} \\ z_{n+1} &\leftarrow z_n - \alpha \nabla_{n+1} \\ x^* &\text{ updated from time to time}\end{aligned}$$

Incorporating Nesterov momentum into Adam
T. Dozat

***Adding gradient noise improves learning
for very deep networks***
A. Neekakantan et al. (2015)

***Exploiting the structure:
stochastic gradient methods using raw clusters***
Z. Allen-Zhu et al. (2016)

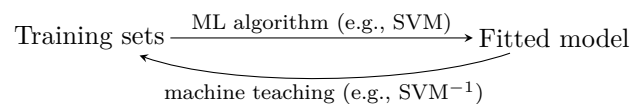
(LSH-based) approximate clustering transforms “big data” into “data”; the clusters can improve variance reduction in SVRG.

Lip reading sentences in the wild
J.S. Chung et al.

Curriculum learning.

***Machine teaching:
an inverse problem to machine learning
and an approach toward optimal education***
X. Zhu

Machine teaching is the inverse problem of machine learning (ML): not going from training data to fitted model, but manufacturing a training dataset for which the ML algorithm of interest would derive the desired model.



Unitary evolution recurrent neural networks
A. Arjovsky et al. (2016)

To avoid exploding or vanishing gradients, use (real) unitary weight matrices (all eigenvalues have modulus 1). Unitary matrices are not easy to parametrize: model them as products of

- (Complex) diagonal matrices;
- Reflexions: $R = I - 2vv^* / \|v\|^2$, $v \in \mathbb{C}^n$;
- Permutation matrices (use a single, fixed one);
- Fourier and inverse Fourier transforms.

**Full-capacity
unitary recurrent neural networks**
S. Wisdom et al. (2016)

Unitary matrices can eliminate the vanishing/exploding gradient problem in RNNs. Instead of using a parametrization of some of those matrices, consider them as points on the (Stiefel) manifold of unitary matrices.

Self-normalizing neural networks
G. Klambauer et al.

Batch/layer/weight normalization may not be needed: scaled exponential linear units (SELU)

$$f(x) = \begin{cases} \lambda x & \text{if } x \geq 0 \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$$

with $\lambda > 1$ (e.g., $\lambda = 1.0507$, $\alpha = 1.7581$) are self-normalizing, in the sense that if $X \sim N(0,1)$ then $Ef(X) = 0$ and $\text{Var } f(X) = 1$ and if X is close to standardized, then $f(X)$ is more so.

**Understanding the difficulty of training
deep feedforward neural networks**
X. Glorot and Y. Bengio (2010)

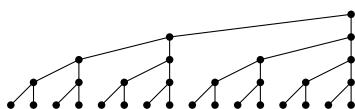
Xavier initialization suggests to select the random initial weights so that the activations of all layers have zero mean and unit variance.

**Modeling temporal dependencies in
high-dimensional sequences: applications to
polyphonic music generation and transcription**
N. Boulanger-Lewandowski et al. (2012)

Restricted Boltzman machines (RBM) economically model high-dimensional discrete distributions. A recurrent temporal RBM (RTRBM) is a sequence of RBMs whose parameters are a (fixed) linear function of the previous hidden state. An RNN-RBM is a sequence of RBMs whose parameters are given by a recurrent neural net (RNN). They can be used as a prior to improve the accuracy of polyphonic transcription.

WaveNet: a generative model for raw audio
A. can den Oord et al.

To generate raw audio samples (16 000 Hz), use dilated causal convolutions, gated units, residual and skip connections.



Strongly-typed recurrent neural networks
D. Balduzzi and M. Ghifary (2016)

Physicists' dimension analysis suggests RNN, LSTM, GRU variants.

**Learning scalable deep kernels
with recurrent structure**
M. Al-Shedivat et al.

Adding a Gaussian process (GP) layer to a recurrent neural net (RNN) and using the negative log-marginal likelihood as objective is problematic: the objective no longer factorizes over the data and stochastic optimization cannot be used – but semi-stochastic alternating gradient descent (full gradient for the GP parameters, stochastic gradient for the RNN) is good enough.

**Professor forcing: a new algorithm
for training recurrent networks**
A. Lamb et al. (2016)

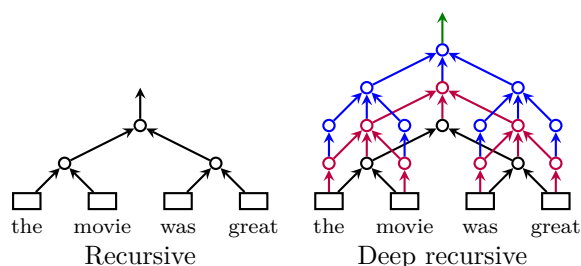
Recurrent neural networks (RNN) generate sequences one character at a time, each conditional on the previous ones. During training, one can use:

- The previous characters of the ground truth sequence (“teacher forcing”, maximum likelihood);
- The previous outputs;
- A mixture of both;
- An adversarial approach, training another model to distinguish between true and generated sequences, and using its performance as a penalty.

**Improved semantic representations
from tree-structured
long short-term memory networks**
K.S. Tai et al.

LSTMs can be generalized from chains to trees.

**Deep recursive neural networks
for compositionality in language**
O. ĩrsoy and C. Cardie



**AtomNet: a deep convolutional neural network
for bioactivity prediction in structure-based
drug discovery**
I. Wallach et al.

CNNs can leverage molecule shapes (on a 3-dimensional grid) for drug discovery.

Unsupervised representation learning with deep convolutional generative adversarial networks
A. Radford et al. (2016)

- Replace pooling layers with strided convolutions (discriminator) or fractionally strided convolutions (generator);
- Remove fully-connected layers;
- Use batch normalization;
- Use ReLU (generator), tanh (generator output) and leaky ReLU (discriminator).

Convolutional networks on graphs for learning molecular fingerprints
D. Duvenaud et al.

To fingerprint molecules (or graphs), for each node, compute a hash of its neighbours, then a hash of the neighbours' hashes, and so on (circular fingerprints). Alternatively, use a graph convnet with large random weights and tanh activations.

Geometric deep learning on graphs and manifolds using mixture model CNNs
F. Monti et al.

CNNs can be defined on arbitrary graphs or manifolds, not just lattices in Euclidian space. For instance, for each vertex x , and each neighbouring vertex $y \in \mathcal{N}(x)$, define pseudo coordinates $\mathbf{u}(x, y)$ (e.g., geodesic polar coordinates (ρ, θ) on a surface, or the degree on a graph), apply a weight function $\mathbf{w}(\mathbf{u}(x, y))$ (e.g., a Gaussian or triangular kernel, possibly with a learnable parameter) and set

$$D_j(x)f = \sum_{y \in \mathcal{N}(x)} \mathbf{w}(\mathbf{u}(x, y))f(y).$$

Graph stream algorithms: a survey
A. McGregor

Many graph problems can be solved, at least approximately, with streaming data, *i.e.*, without storing the whole graph in memory (but you still need $\Omega(\#V)$ storage):

- Test connectivity by building a spanning forest (a set of edges): add the next edge $\{u, v\}$ if there is currently no path from u to v ;
- Approximate distances by building a subgraph: add the next edge $\{u, v\}$ if the current $d(u, v)$ is above some threshold (*i.e.*, if the new edge closes a small loop);
- Minimum spanning tree (MST);
- Approximately count triangles by considering the vectors with coordinates $x_T = \# \text{edges in } T$ where $T \in \mathcal{P}_3(V)$;
- Greedy matching;
- Random walks (\sqrt{t} passes on the data suffice: stitch \sqrt{t} walks of length \sqrt{t}).

Many streaming algorithms use a *linear sketch*, *i.e.*, a random projection (of the adjacency or some related

matrix). Algorithms for sliding window graphs (the last w edges in an infinite stream) often keep the k most recent edges satisfying some property.

Darwini: generating realistic large-scale social graphs
S. Edunov et al.

To generate a (large) graph with prescribed degree and clustering coefficient distributions (conditional on degree):

- Assign a target degree and clustering coefficient to each node;
- Group the nodes in clusters, according to $cd(d-1)$;
- Build an Erdős-Rényi graph on each cluster, choosing the probability (and cluster size) to get the desired expected clustering coefficient (and a degree inferior to the target);
- Add edges between clusters to get the desired degree distribution [tricky for large-degree nodes], trying to keep the joint degree distribution.

How to partition a billion-node graph
L. Wang et al. (2014)

To partition a graph (for distributed processing, or sparse matrix ordering):

- Coarsen it by finding a maximum match (a set of edges sharing no vertices) and collapsing its edges, until the graph is sufficiently small;
- Assign a unique label to each vertex; set the label of each vertex to the majority label in its neighbourhood;
- If there are too many labels, collapse vertices with the same label, and propagate labels anew.

Graphons, mergeons and so on
J. Eldridge et al. (2016)

A **graphon** is a symmetric measurable function $w : [0, 1]^2 \rightarrow [0, 1]$. It can be seen as a weighted graph on the uncountable vertex set $[0, 1]$. It defines a sequence of random graphs (graph-valued random variable) $(G_n)_n$, where G_n samples n points x_i uniformly in $[0, 1]$ and then an edge $i-j$ with probability $w(x_i, x_j)$. It is *consistent* (the random graph obtained by removing the $(n+1)$ st vertex from G_{n+1} has the same distribution as G_n) and *local* (if $S, T \subset [n]$ are disjoint, then $G_n|_S \perp\!\!\!\perp G_n|_T$); these properties characterize graphons. A subset $A \subset [0, 1]$ is *disconnected* at level λ if there exists $S \subset A$ such that $0 < \mu(S) < \mu(A)$ and $w < \lambda$ on $S \times (A \setminus S)$. Define an equivalence relation \sim on the set \mathcal{A}_λ of λ -connected subsets by $A_1 \sim A_2$ iff $\exists A \in \mathcal{A}_\lambda \quad A \supset A_1 \cup A_2$. The clusters at level λ are the (essential) largest elements in each equivalence class. A **mergeon** describes how clusters merge as λ decreases: it is a graphon M such that

$$M^{-1}[\lambda, 1] = \bigcup_{C \text{ } \lambda\text{-cluster}} C \times C.$$

GraphChi-DB: simple design for a scalable graph database system – on just a PC

A. Kyrola and C. Guestrin

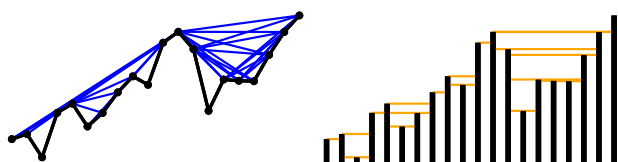
Adapt the *log-structured merge tree* (LSM tree) to store edges: partition them on the destination vertex, but sort each partition along the source vertex.

Time series analysis of the S&P 500 index: a horizontal visibility graph approach

M.D. Vamvakaris et al. (2017)

The *horizontal visibility graph* is a variant of the visibility graph:

- Fit the degree distribution with a power law: if the exponent is greater than $\log 3/2$, the time series is auto-correlated; if it is less, it is chaotic;
- The *Hellinger distance* between the in- and out-degree distributions measures irreversibility.



Process systems engineering as a modeling paradigm for analyzing systemic risk in financial networks

R. Bookstaber et al. (2015)

The financial system can be modeled as a *signed-directed graph* (SDG): the directed arcs represent causal relations, and the sign of $x \rightarrow y$ is that of $\partial y / \partial x$. *Process hazard analysis* can help identify feedback loops, which amplify shocks – many agents act in a locally stabilizing but globally destabilizing way.

Diffusion-convolutional neural networks

J. Atwood and D. Towsley (2016)

On a graph, one can forecast node labels Y from node features X by using the $A^n X$ as predictors, where A is the adjacency matrix. Averaging over the nodes, this can be generalized to edge or graph labels.

A machine learning perspective on predictive coding with PAQ

B. Knoll and N. de Freitas (2011)

Prediction by partial matching (PPM) is a lossless compression algorithm well-suited to text:

- Start with a prior probability distribution on the letters;
- Encode the first character with the Huffman code or the arithmetic code for this distribution;
- Somehow update the probability distribution; iterate.

For instance, one could use an n -gram model, *i.e.*, keep track of all the substrings seen so far, and use

the longest matches to predict (the probability) of the next letter. With noisy inputs, approximate matching helps. For images, scan by row or column, or along a space-filling curve (Hilbert) – you can beat JPEG but not JPEG2000.

PAQ8 uses 500 such models, for bit-level predictions, ensembled with a *mixture of experts* (implemented as a neural net).

Compression can be used to define distances

$$d_C(x, y) = \frac{C(x|y) + C(y|x)}{C(xy)}$$

$$d_{\text{CDM}}(x, y) = \frac{C(xy)}{C(x) + C(y)}$$

$$d_e(x, y) = \frac{1}{2} [E[x|y] + E[y|x]]$$

$$d_{\text{NDM}}(x, y) = \frac{C(x, y) - \text{Min}\{C(x), C(y)\}}{\text{Max}\{C(x), C(y)\}}$$

where $C(x)$ is the compressed size of x (approximation of its Kolmogorov complexity), $C(x|y)$ is the compressed size of x if the compressor is trained on y , $E[x|y]$ is the corresponding cross-entropy of x .

Arithmetic coding encodes a text given a sequence of probability distributions on letters, as a single rational number: divide the interval $[0, 1]$ into segments, one for each possible letter, of length their probabilities; pick the segment corresponding to the current letter; repeat the process on that segment, with the next letter; iterate and return any rational number in the final segment.

A brief review of the ChaLearn AutoML challenge

I. Guyon et al. (2016)

AutoML is a competition for automated (black-box, with no human intervention) supervised learning, combining the following ideas:

- Decision trees, k -NN, naive Bayes, neural nets;
- Uniformize, replace missing values, group modalities of categorical variables, feature selection, dimension reduction (PCA, kPCA, ICA, MDS, LLE, Laplacian eigenmaps), clustering (k -means)
- Hyperparameter tuning (SMAC, hyperopt)
- Model selection (k -fold cross-validation, l.o.o, o.o.b., bilevel optimization).

Structure discovery in nonparametric regression through compositional kernel search

D. Duvenaud et al. (2013)

The “automated statistician” performs a search in a space of kernel structures, built from sums and products of linear, square exponential, periodic, etc. bricks, in a Gaussian process.

**Machine learning techniques: reductions
between prediction quality metrics**
A. Beygelzimmer et al.

Reduction is the set of techniques converting a loss-minimization problem into a well-studied machine-learning one, typically binary classification or regression. Examples include:

- Importance-weighted classification (different cost for false positives and negatives) by reweighting the training set;
- Multiclass classification with one-against-all (inconsistent), error correcting codes (ECOC, inconsistent), probabilistic ECOC, all-pairs;
- Cost-sensitive multiclass classification, with weighted all-pairs;
- Quantile regression with importance weighted classification;
- Ranking (of binary data).

Multi-view machines
B. Cao et al.

Sparse variant of factorization machines.

**FastDBT: A speed-optimized
and cache-friendly implementation
of stochastic gradient-boosted decision trees
for multivariate classification**
T. Keck

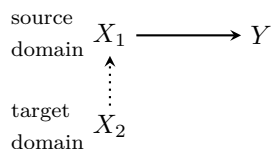
May be faster than XGBoost. Also check LightGBM and CatBoost.

**Coresets for scalable
Bayesian logistic regression**
J.H. Higgins et al. (2016)

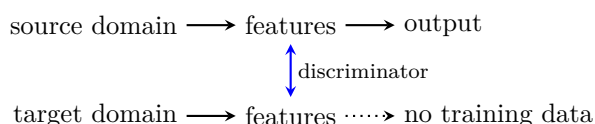
Replace large datasets with a *coreset*, i.e., a weighted (smaller) subset (obtained, e.g., from an approximate streaming clustering algorithm) to approximate Bayesian posterior likelihood (for logistic regression).

**Domain-adversarial training
of neural networks**
Y. Ganin et al. (2016)

Domain adaptation (DA) is the transfer of a model from one domain to another, sometimes by a mapping (alignment) between the two domains.

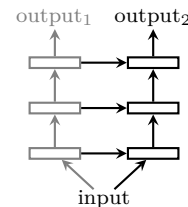


One can also use an adversarial approach to build domain-invariant features.



Progressive neural networks
A.A. Rusu et al.

To learn a task related to an already-learned one, consider a new neural net, parallel to the first one, initialized with random weights, with lateral connections. To keep the model scalable (task $n + 1$ has connections from tasks $1, 2, \dots, n$), use dimension reduction (single-layer network) for those lateral connections.



**Net2Net: accelerating learning
via knowledge transfer**
T. Chen et al. (2016)

Transform an already trained network into a deeper one by inserting a layer initialized as the identity function, or into a wider one by duplicating some of the nodes (keep their input weights and halve their output weights).

Sampling generative networks
T. White (2016)

In high dimensions, linear interpolation goes through atypical points (too close to the origin): prefer *spherical interpolation*.

To study analogies, e.g., king – man + woman, generate images in a grid having those three points as corners.

In the latent space, find the directions corresponding to “smile”, “open mouth”, “gender” or even “blurry” and generate a 1- or 2-dimensional grid of images by moving along those directions.

**Efficient convolutional auto-encoding
via random verification
and frequency-domain minimization**
M.C. Oventneke et al. (2016)

Use random projections for the non-linear encoding part of the auto-encoders, and only learn the (linear) decoding part. [cf echo networks, reservoir learning]

Universal adversarial perturbations
S.M. Moosavi-Dezfooli et al.

There exist almost universal adversarial perturbations: a small perturbation which, when added to almost any image, tricks a neural network into misclassifying it.

Matrix neural networks
J. Gao et al. (2016)

If your neural network has matrices as input (instead of vectors), try *bilinear layers*: $Y = \sigma(UXV' + B)$ (U , V and B are matrices).

**FastText.zip:
compressing text classification models**
A. Joulin et al.

Product quantization decomposes the space into k orthogonal subspaces $\mathbf{R}^n = \bigoplus_{i=1}^k V_i$, computes 2^b (with $b = 8$) k -means centroids in each of them, and approximates points as sums of those centroids $x \approx \sum q_i(x)$.

Use when you want to approximate scalar products – e.g., for vector embeddings.

Tensorizing neural networks
A. Novikov et al.

Dimension reduction (TensorTrain) for dense weight matrices (of fully-connected layers).

**Neural network based clustering
using pairwise constraints**
Y.C. Hsu and Z. Kira (2016)

Cluster data by training a (softmax) network on similar and dissimilar *pairs*.

**Composing graphical models
with neural networks for structured
representations and fast inference**
M.J. Johnson et al.

Use neural nets to add non-linear transformations to your graphical models; inference is tricky: SVAE (structural variational encoders) use stochastic gradients of a variational approximation.

**mgm: Structure estimation
for time-varying mixed graphical models
in high-dimensional data**
J.M.B. Haslbeck and L.J. Waldorp

To estimate a Gaussian graphical model, identify the neighbourhood of each node with a lasso regression. For mixed data (exponential family), use GLMs. For time-varying models, use (Gaussian) weights.

**Inference compilation
and universal probabilistic programming**
T.A. Le et al.

Neural networks can be trained to provide better proposal distributions for sequential importance sampling (SIS).

Memory efficient kernel approximation
S. Si et al. (2014)

Approximate large kernel matrices as a direct sum of low-rank matrices, where the blocks correspond to clusters.

**Multi-class generative adversarial networks
with the L^2 loss function**
X. Mao et al.

Replace the sigmoid cross-entropy loss in the discriminator with an L^2 loss.

**GANs for sequences of discrete elements
with Gumbel-softmax distribution**
M. Kusner and J.M. Hernández-Lobato

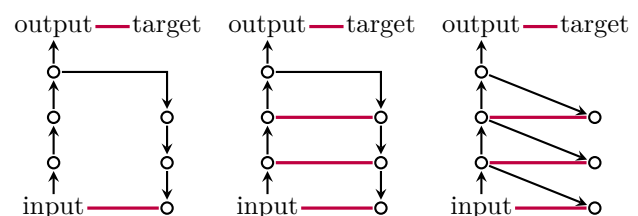
The Gumbel softmax distribution is the distribution of

$$y = \text{softmax} \frac{1}{\tau(h + g)}$$

where $g_i \stackrel{\text{iid}}{\sim} \text{Gumbel}(0, 1)$. When $\tau \rightarrow 0$, this becomes the one-hot encoding of $h_i + g_i$, i.e., $y \sim \text{Multinomial}(p)$, where $p = \text{softmax } h$, i.e., $p_i \propto \exp h_i$. It can be used as a differentiable approximation of the multinomial distribution, e.g., in GANs.

**Augmenting supervised neural networks
with unsupervised objectives
for large-scale image classification**
Y. Zhang et al. (2016)

Augment supervised networks with autoencoder losses.



Infinite-dimensional word embedding
E.T. Nalisnick and S. Ravi (2016)

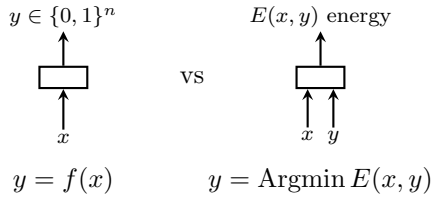
Make word embeddings “infinite dimensional” by initializing them with high-dimensional (not really infinite) vectors, using only the first z components, with a penalty for z , and a sparsity penalty on the vectors [unrelated to Dirichlet processes, iHMM and other finite-dimensional models].

Pointer networks
O. Vinyals et al.

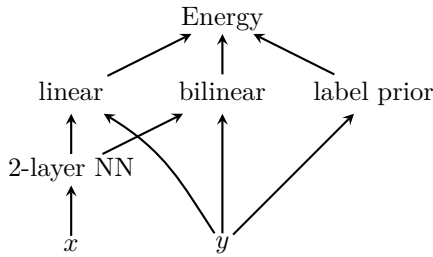
Attention-based networks for TSP, convex hull, Delaunay triangulation.

Structured prediction energy networks
D. Belanger and A. McCallum (2016)

Structured prediction can be performed indirectly, by learning an energy function and minimizing it.



For instance, the model could use a 2-layer neural net to compute features and combine them, linearly and bilinearly, with the label. The problem can be relaxed from $y \in \{0, 1\}^n$ to $y \in [0, 1]^n$.



To learn the model, notice that we want $\forall y E(x_i, y) \geq E(x_i, y_i)$, with a larger difference if y and y_i are far apart, e.g., $\forall y E(x_i, y) - E(x_i, y_i) \geq d(y, y_i) \geq 0$. This suggests minimizing a *structured SVM* (SSVM) loss

$$\sum_i \text{Max}_y [d(y, y_i) - E(x_i, y) + E(x_i, y_i)]_+.$$

Machine learning techniques for Stackelberg security games: a survey
G. De Nittis and F. Trovò (2016)

In the Stackelberg security game (SSG), a defender protects T targets from an attacker, by allocating resources $R < T$. The strategies are $\{x \in \mathbf{R}^T : 0 \leq x \leq 1, x' \mathbf{1} \leq R\}$. If the adversary attacks t , both receive a penalty and a reward:

$$\begin{aligned} \text{Attacker} &: x_t P_t^a + (1 - x_t) R_t^a \\ \text{Defender} &: (1 - x_t) P_t^d + x_t R_t^d \end{aligned}$$

Boundedly rational adversaries attack a target at random, with a probability depending on their (potentially incorrect) subjective utility. The subjective utility function can be estimated from data (poaching and illegal fishing provide real-world data). The defender's strategy should be robust to suboptimal adversaries.

Apprenticeship learning using inverse reinforcement learning and gradient methods
G. Neu et al. (2007)

Inverse reinforcement learning (IRL) learns a policy (a map from state features to actions) from an expert's observed behaviour. Instead, one can learn the

reward function for which the optimal policy is as close to the expert's as possible.

Dueling network architectures for deep reinforcement learning
Z. Wang et al.

In deep reinforcement learning, separately learn the state value function and the state-dependent action advantage function.

Deep successor reinforcement learning
T.D. Kulkarni et al.

Successor reinforcement learning learns the value function as the inner product between a reward predictor and a successor map.

Convexified convolutional neural networks
Y. Zhang et al. (2016)

Training a 1-hidden-layer CNN with linear activations is an optimization problem with a low-rank constraint (corresponding to weight sharing) which can be relaxed into a nuclear norm constraint. For non-linear activations, use a kernel (RKHS). For deeper networks, this no longer works, but greedy layer-wise training seems a good heuristic.

End-to-end kernel learning with supervised convolutional kernel networks
J. Mairal (2016)

A *convolutional kernel network* uses the kernel trick to increase the dimension and add non-linearities

$$\mathbf{R}^n \xrightarrow{\phi} \mathbf{R}^N \xrightarrow{\text{projection}} \mathbf{R}^m, \quad N \gg n, m$$

without explicitly computing the high-dimensional coordinates. The kernel parameters can be learnt as usual.

Network in network
M. Lin et al.

CNNs slide simple masks over an image, to detect features. Instead, one could slide slightly deeper networks ("micro-networks").

Towards a mathematical theory of super-resolution
E.J. Candès and C. Fernandez-Granda (2012)

It is possible to recover a spike train (*i.e.*, a discrete complex measure, $x = \sum_i a_i \delta_{t_i}$) from low-resolution measurements or, equivalently, from a truncated Fourier transform $\hat{x} = \mathcal{F}_n x$, by solving a convex problem

$$\begin{aligned} \text{Find} & \quad y, \text{ a complex measure} \\ \text{To minimize} & \quad \|y\|_{\text{TV}} \\ \text{Such that} & \quad \mathcal{F}_n y = \hat{x} \end{aligned}$$

provided the spikes are sufficiently separated, where the total variance of a measure μ on $[0, 1]$ is a continuous analogue of the ℓ^1 norm:

$$\|\mu\|_{TV} := |\mu|([0, 1]).$$

Multiscale Retinex
A.B. Petro et al. (2014)

The *retinex* algorithm is a local contrast enhancement algorithm. Consider random walks from a point, x_0, x_1, \dots, x_n ; compute

$$L = \sum \delta \left(\log \frac{I(x_k)}{I(x_{k+1})} \right),$$

where $\delta(u) = u \mathbf{1}_{|u| \geq t}$ is a thresholding function, and use $\langle L \rangle$ as the new luminance.

Alternatively, use the ratio of a pixel value to the average of its neighbours (with Gaussian weights); repeat for several scales ($\sigma = 15, 80, 250$), and average. Restore the colours to their original range, or apply the algorithm to the intensity channel only.

A point set generation network for 3D object reconstruction from a single image
H. Fan et al.

To reconstruct a 3D scene as a cloud of points (union of balls – this eliminates overlap and connectivity problems), learn a neural net consuming an image and random data. The resulting shape sampler also accounts for ground truth ambiguity.

Spatial transformer networks
M. Jaderberg et al.

Add a transformation layer, to help align images – it is differentiable, so it will go through gradient descent.

StackGAN: text to photorealistic image synthesis with stacked generative adversarial networks
H. Zhang et al.

Stack several GANs, generating images of increasing resolution.

The more you know: using knowledge graphs for image classification
K. Marino et al.

LSTMs can be generalized to arbitrary DAGs (instead of time, $\bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet \rightarrow \bullet$).

Propagating over the “promising” subset of the graph may be sufficient.

A deep and autoregressive approach for topic modeling of multimodal data
Y. Zheng et al. (2016)

Represent images as bags of visual words (e.g., SIFT features) to use topic models (LDA, DocNADE).

v_1, \dots, v_d words

$p(\mathbf{v}) = \prod p(v_i | \mathbf{v}_{<i})$ or a balanced binary tree

$$\mathbf{h}_i(\mathbf{v}_{<i}) = g \left(c + \sum_{k < i} W_{\bullet, v_k} \right)$$

$$p(v_i = w | \mathbf{v}_{<i}) \propto \exp(b_w + V_{w, \bullet} \mathbf{h}_i(\mathbf{v}_{<i}))$$

Simplicial mixtures of Markov chains: distributed modelling of dynamic user profiles
M. Girolami and A. Kabán

Variant of latent Dirichlet allocation (LDA: a document is a mixture of topics, a topic is a distribution on words) for mixtures of Markov chains: user activity is a mixture of behaviours, a behaviour is a distribution on sequences (Markov chains).

Solving heterogeneous estimating equations with gradient forests
S. Athey et al. (2016)

Random forests are (high-dimensional) adaptive nearest neighbour functions: they can be used as an alternative to kernel-based weighting, where the weight of an observation is the proportion of trees in which it is in the same leaf as the point of interest. The choice of the splits can be tailored to the model (quantile regression, etc.) by minimizing the after-cut squared error, or some gradient-based approximation.

Implementation in `gradientForest`, `ranger`.

bartMachine: machine learning with Bayesian additive regression trees
A. Kapelner and J. Bleich

Forests of regression (not classification) trees, with a prior on tree structure and leaf parameters:

- Nodes at depth d are non-terminal with probability $\alpha(1+d)^{-\beta}$, $\alpha = .95$, $\beta = 2$,
- Splitting variables are selected uniformly at random,
- Splitting values are sampled from the multiset of values taken,
- The parameters are Gaussian,
- The noise is inverse Gamma,

estimated with Metropolis-within-Gibbs sampling (Metropolis for the tree structure, Gibbs for the rest), in Java.

On the generalized ratio of uniforms as a combination of transformed rejection and extended inverse of density sampling
L. Martino et al.

Let p be a monotonic probability density (e.g., a half-Gaussian) and

$$\mathcal{A} = \left\{ (u, v) \in \mathbf{R}^2 : 0 \leq u \leq \sqrt{p(v/u)} \right\},$$

If $(U, V) \sim \text{Unif } \mathcal{A}$, then $V/U \sim p$.

This can be generalized: let $g : \mathbf{R}_+ \rightarrow \mathbf{R}$, \mathcal{C}^1 , strictly increasing, $g(0) = 0$,

$$\mathcal{A}_g = \left\{ (u, v) \in \mathbf{R}^2 : 0 \leq u \leq g^{-1} \left[c \cdot p \left(\frac{v}{g(u)} \right) \right] \right\},$$

if $(U, V) \sim \text{Unif } \mathcal{A}_g$, then $V/g(U) \sim p$.

Inverse of density sampling, for monotonic probability density functions, uses

$$(X, Y) \sim \text{Unif} \{ (x, y) : 0 \leq y \leq p(x) \}$$

or, if it is easier to sample from p^{-1} ,

$$Y \sim p^{-1}, \quad X \sim \text{Unif}[0, p^{-1}Y].$$

In both cases, $X \sim p$.

For non-monotonic densities, decompose them into intervals on which they are either increasing or decreasing.

Pareto smoothed importance sampling
A. Vehtari et al. (2016)

Importance sampling estimates

$$\int f dP = \int f \frac{dP}{dQ} dQ \approx \frac{1}{N} \sum f(x_i) \frac{p(x_i)}{q(x_i)},$$

where $x_i \sim Q$, when it is easier to sample from Q than P . If p is only known up to a normalization factor,

$$\int f dP \approx \frac{\sum f(x_i) \frac{p(x_i)}{q(x_i)}}{\sum \frac{p(x_i)}{q(x_i)}}.$$

To stabilize this ratio estimate, one can truncate the weights $w_i = p(x_i)/q(x_i)$, but this introduces bias. Alternatively, one can fit a Pareto distribution to their tail and replace the tail values with the corresponding Pareto quantiles.

Constrained maximum correntropy adaptive filtering
S. Peng et al. (2016)

Maximum correntropy estimators use a kernel as a loss function, e.g.,

$$\begin{aligned} \text{Loss}(\hat{y}, y) &= - \sum \kappa(\hat{y}, y) \\ \kappa(x, y) &= \exp - \frac{1}{2} \left(\frac{x - y}{\sigma} \right)^2 \end{aligned}$$

Learning design patterns with Bayesian grammar induction
J.O Talton et al. (2012)

It is not possible to learn a (deterministic) grammar from positive examples, but it is possible for probabilistic languages (grammar induction). A Bayesian approach would be:

- Build a grammar that recognizes all exemplars and nothing else;
- Modify it (MCMC), to reduce its length while keeping the likelihood high.

A case for robust Bayesian priors with applications to binary clinical trials
J.A. Fúquene et al.

Conjugate Bayesian analysis is not robust to the prior. The Cauchy prior (for the normal log-odds model, or the binomial model for binary data) is robust but does not lead to a closed form posterior. *Berger's prior* (a continuous mixture of Gaussians) is also robust and does.

RUSBoost: a hybrid approach to alleviating class imbalance
C. Seiffert et al. (2010)

Class imbalance can be addressed by:

- Adding more samples to the minority class: over-sampling, boosting, extrapolating (SMOTE);
- Removing samples from the majority class: under-sampling (RUS).

Those methods can be combined.

Orthant probabilities for robust correlation and structural performance enhancement
M. Anderson et al. (2015)

To capture non-linear effects with correlations, notice that if (X, Y) is Gaussian

$$\begin{aligned} P_{ij} &= P[(-1)^i X > 0 \wedge (-1)^j Y > 0] \quad i, j \in \{0, 1\} \\ &= \frac{1}{4} + \frac{(-1)^{i+j}}{2\pi} \arcsin \rho \end{aligned}$$

and convert those orthant probabilities into 4 **quadrant correlations**.

Efficient thresholded correlation using truncated singular value decomposition
J. Baglama et al. (2016)

To find all pairs of variables (X_i, X_j) with correlation above a threshold $0 < t < 1$ without examining them all, notice that, if $\forall i \|x_i\| = 1$,

$$\|x_i - x_j\|^2 > 2(1 - t) \implies \text{Cor}(x_i, x_j) < t$$

and $\|Px_i - Px_j\| \leq \|x_i - x_j\|$ if P is a projection. Only examine pairs of vectors whose projections on the first few singular vectors are sufficiently close.

A survey of discretization techniques: taxonomy and empirical analysis in supervised learning
S. García et al. (2013)

Comparison of 80 discretization algorithms (equal width, equal frequency, χ^2 , etc.), without any detail.

Efficient thresholded correlation using truncated singular value decomposition
J. Baglama et al. (2016)

To find the most correlated (*i.e.*, the closest) pairs of unit vectors, project them on the first singular value dimensions.

Model selection for Gaussian process regression by approximation set coding
B. Fischer et al.

Approximation set coding (ASC) is a model selection procedure maximizing the posterior agreement η_θ

$$\begin{aligned}\theta &: \text{hyperparameters} \\ \alpha &: \text{parameters} \\ p_\theta(\alpha) &: \text{prior} \\ D &= D_1 \sqcup D_2 \text{ random partition of the data} \\ \eta_\theta &= \int p_\theta(\alpha|D_1)p_\theta(\alpha|D_2)p_\theta(\alpha)d\alpha.\end{aligned}$$

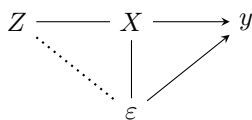
The likelihood can be:

$$\begin{aligned}p(\alpha|D) &\propto p_\theta(D|\alpha)p_\theta(\alpha) && \text{Bayesian} \\ (\alpha|D) &\propto p_\theta(D|\alpha)^\beta && \text{Entropy } (\beta = 1).\end{aligned}$$

It can be used to select the kernel of a Gaussian process: square exponential e^{-x^2} , rational quadratic $(1+x^2)^{-\alpha}$, exponential $e^{-|x|}$, periodic $e^{-\sin^2 x}$.

Counterfactual prediction with deep instrumental variable networks
J. Hartford et al.

Instrumental variables can be added to deep neural networks.



Doubly stochastic neighbor embedding on spheres
Y. Lu et al.

Normalizing the t-SNE similarity matrix to make it doubly stochastic limits the crowding around the origin. The resulting approximately spherical embedding can be plotted on \mathbf{S}^2 .

Causal inference and the data-fusion problem
E. Bareinboim and J. Pearl (2016)

The rules of *do-calculus* (for causal inference) can be expressed in terms of conditional independence and graphical models.

Principled detection of out-of-distribution examples in neural networks
S. Liang et al.

To help detect out-of-distribution observations, increase the temperature of the softmax

$$p_i(x) \propto \exp \frac{f_i(x)}{T}$$

and add noise $x \leftarrow x - \varepsilon \cdot \text{sign} - \nabla_x \log p_{\hat{y}}(x)$.

HiCS: high contrast subspaces for density-based outlier ranking
F. Keller et al.

Outlier detection algorithms often compare the density of a point with that of its neighbours – they suffer from the curse of dimensionality. Instead, compute the LOF (local outlier factor) for high-contrast subspaces, defined as axis-aligned subspaces on which the conditional distributions $Y|X = x$ (estimated on slices $Y|x_1 \leq Xx_2$) and the marginal distribution Y are significantly different, as measured by a Welsh test (T test with unequal variances) or a Kolmogorov-Smirnov test.

Programming with models: writing statistical algorithms for general model structures with Nimble
P. de Valpine et al.

Nimble lets you describe a computational graph in R, and compiles it into C++, for use in Bugs; it also lets you change how the data is sampled (importance sampling, Metropolis Hastings).

sgof: an R package for multiple testing problems
O. Castro-Conde and J. de Uña-Álvarez

For a large number of tests, SGoF (sequential goodness of fit, which compares $\#\{p : p < \gamma\}$ with its expected value γ under H_0) is more powerful than FDR or FWER approaches. Also check: `qvalue` (the q-value is the proportion of false positives), `HybridMTest`, `multtest`, `mutoss`, `multcomp`, `stats::p.adjust`.

conting: an R package for Bayesian analysis of complete and incomplete contingency tables
A.M. Overstall and R. King (2014)

Log-linear models, for contingency tables, are of the form

$$\begin{aligned}y_1 &\sim \text{Poisson } \mu_i \\ \mu_i &= x'_i \theta\end{aligned}$$

where i is a multiindex and \mathbf{x} a design matrix describing which interactions the model allows. For instance, $\mu_{ijk} = a + b + i + c_j + d_k$ or $\mu_{ijk} = a + b + i + c_j + d_k + e_{ij} + f_{ik} + g_{jk}$. In a Bayesian context, the model \mathbf{x} is also unknown. The *generalized hyper-g-prior* for $\theta|x$ is the posterior from a locally uniform prior and an imaginary sample of size $1/\sigma^2$.

**Counterfactual:
an R package for counterfactual analysis**
M. Chen et al.

Given two populations 0 and 1 (e.g., men and women), counterfactual analysis tests if the differences in the distributions of the same variable y (e.g., salary) comes from different characteristics x (are the marginal distributions x the same?) or a different treatment (are the conditional distributions $y|x$ the same?).

$$F_{\langle ij \rangle}(y) = \int_{\mathcal{X}_j} F_{Y_i|X_j}(y|x) dF_{X_j}(x), \quad i, j \in \{0, 1\}$$

$$F_{\langle 11 \rangle}^{-1} - F_{\langle 00 \rangle}^{-1} = (F_{\langle 11 \rangle}^{-1} - F_{\langle 01 \rangle}^{-1}) + (F_{\langle 01 \rangle}^{-1} - F_{\langle 00 \rangle}^{-1})$$

Multilabel classification with R package mlr
P. Probst et al.

Multiple label classification assigns several labels (or “tags”) to each observation. One could train a classifier for each label, sepeartely, but this assumes they are independent. Instead, one can add all or some of the labels, either the true or forecasted values, to the predictors.

**coxphMIC: an R package for sparse
estimation of Cox proportional hazards models
via approximate information criteria**
R. Nabi and X. Su

Best subset selection models add a penalty of the form $\lambda \sum \mathbf{1}_{\beta_i \neq 0}$, with $\lambda = 2$ (AIC) or $\lambda = \log n$ (BIC). It can be relaxed by replacing $\mathbf{1}_{\beta \neq 0}$ with $\tanh \beta^2$ (or $\tanh(a\beta^2)$), but since it is smooth in 0, it does not give sparse solutions. Instead, replace $\beta = \beta \mathbf{1}_{\beta \neq 0}$ with $\beta = \gamma \tanh \gamma^2$ and use $\tanh^2 \gamma$ as a penalty.



**GsymPoint: an R package to estimate
the generalized symmetry point,
an optimal cut-off point
for binary classification
in continuous diagnostic tests**
M. López-Ratón et al.

To find the best cutoff point on a ROC curve, one can use the *symmetric point*, i.e., the intersection with the diagonal $y = 1 - x$, if the costs of false positives and negatives are the same or, more generally, the solution of $\rho(1 - \text{specificity}) = 1 - \text{sensitivity}$, where

$$\rho = \frac{\text{cost FP}}{\text{cost FN}},$$

which corresponds to $E[\text{cost FP}] = E[\text{cost FN}]$. If the scores are Gaussian,

$$Y_{\text{healthy}} \sim N(\mu_0, \sigma_0)$$

$$Y_{\text{diseased}} \sim N(\mu_1, \sigma_1),$$

then $\text{ROC}(x) = \Phi(a + b\Phi^{-1}(x))$, $a = (\mu_1 - \mu_0)/\sigma_1 > 0$, $b = \sigma_0/\sigma_1$. If there is uncertainty on μ_i , σ_i , ρ , one can also compute confidence intervals for the generalized ($\rho \neq 1$) symmetry point.

**spcadjust: an R package for adjusting
for estimation error in control charts**
A. Gandy and T. Kvaløy

Use the bootstrap to estimate the threshold to use in *control charts* to spot a move from $N(\mu, \sigma^2)$ to $N(\mu + \Delta, \sigma^2)$.

$$\text{Shewhart} \quad S_t = \frac{X_t - \hat{\mu}}{\hat{\sigma}}$$

$$\text{CUSUM} \quad S_t = \text{Max} \left\{ 0, S_{t-1} + \frac{X_t - \mu - \frac{1}{2}\Delta}{\sigma} \right\}$$

**Weighted effect coding
for observational data with wev**
R. Nieuwenhuis et al.

With unbalanced classes, consider *weighted effects* to put categorical variables into the design matrix.

**IsoGeneGUI: multiple approaches
for dose-response analysis
of microarray data using R**
M. Otava et al.

Isotonic regression estimates means μ_1, \dots, μ_n under a monotonicity constraint. The corresponding test is $H_0: \forall i \neq j \mu_i = \mu_j$ versus $H_1: \forall i < j \mu_i \leq \mu_j$ or $H_1: \forall i < j \mu_i \geq \mu_j$. The constraint can also be applied to clustering (e.g., by using the isotonic means instead of the raw data).

**milr: multiple-instance logistic regression
with Lasso penalty**
P.Y. Chen et al.

Multiple instance learning considers bags of observations rather than observations:

$$i : \text{bag of instances}$$

$$(i, j) : \text{instance}$$

$$x_{ij} : \text{covariates}$$

$$p_{ij} = g(x'_{ij}\beta) \text{ defect rate of instance } (i, j)$$

$$Y_{ij} \sim \text{Bernoulli}(p_{ij})$$

$$\pi_i : \text{defect rate of bag } i$$

$$\pi_i \propto \sum_j p_{ij} e^{ap_{ij}} \text{ or } 1 - \prod_j (1 - p_{ij})$$

$$\text{or } h\left(\sum h(Y_{ij} = 1)\right).$$

***phtt: Panel data analysis
with heterogeneous time trends in R***
O. Baba and D. Liebl (2014)

Panel models are usually of the form

$$y_{it} = \sum_j x_{ijt} \beta_j + \nu_t + \varepsilon_{it}.$$

Big data allows the time-varying component ν_t to be subject-specific and described by a factor model

$$y_{it} = \sum_j x_{ijt} \beta_j + \nu_{it} + \varepsilon_{it}$$

$$\nu_{it} = \sum_\ell \lambda_{i\ell} f_{\ell t};$$

neither λ nor f is observed, the $f_{\bullet t}$ are orthonormal, so are the $\lambda_{i\bullet}$. The factors can be assumed to be smooth (splines, KSS) or ARIMA (Eup)

***Bayesian state-space modelling
on high-performance hardware using LibBi***
L.M. Murray

LibBi is a BUGS-like language for parallel (multicore CPU, GPU) Bayesian estimation of state space models, using particle filters (sequential Monte Carlo, SMC) instead of Gibbs sampling (as Jags) or Hamiltonian Monte Carlo (HMC, as Stan).

***GPflow: a Gaussian process library
using TensorFlow***
A.G.G. Matthews et al.

Also check: GPy (Python), GPML (Matlab), GPstuff (Matlab).

***Visualizing dependence
in high-dimensional data:
an application to S&P 500 constituent data***
M. Hofert and W. Oldford (2016)

Scatterplot matrices contain each plot twice; it is possible to show each plot only once, while ensuring that each plot shares its vertical (resp. horizontal) axis with a neighbour. This is implemented in the `zenplot` package.

Given a measure of interestingness, one can reduce the number of plots further, and arrange them along an *Eulerian path* (PairViz package).

This can be applied to stock returns, de-GARCHed (ARMA-GARCH residuals, from `qrmtools::fit_ARMA_GARCH`, `rugarch::ugarchfit`) and their tail dependence indices (`copula::fitCopula(tCopula..., method="itau.mpl")`).

***Probabilistic data analysis
with probabilistic programming***
F. Saad and V. Mansinghka

Composable generative population models (CGPM) are a generalization of CrossCat for probabilistic modeling in BayesDB (and its Bayesian query language,

BQL); they can be written VentureScript (a probabilistic programming language) or even Stan.

***Edward: a library for probabilistic modeling,
inference and criticism***
D. Tran et al. (2016)

Edward is a Python probabilistic programming library, built on TensorFlow, based on *variational inference*: describe the model, the priors, a family of posterior approximations, and find the posterior approximation minimizing the KL divergence with the true posterior – this also includes MAP (the posterior approximation of a point mass) and MCMC (an empirical distribution).

***RankPL: a qualitative
probabilistic programming language***
T. Rienstra

Spohn's δ rank theory is a qualitative alternative to probability theory, replacing probabilities with degrees of surprise (ranks). A ranking function $\kappa : \Omega \rightarrow \mathbf{N} \cup \{\infty\}$ is extended to $\mathcal{P}(\Omega)$ as $\kappa(A) = \text{Min}\{\kappa(a), a \in A\}$. (One usually assumes $\kappa(\Omega) = 0$: replace κ with $A \mapsto \kappa(A) - \kappa(\Omega)$ if needed.) The conditional rank is $\kappa(A|B) = \kappa(A \cap B) - \kappa(B)$ if $\kappa(B) \neq \infty$. There is a formal similarity:

Probability	0	1	+	/	×
Rank	∞	0	Min	–	+

TRX: A formally verified parser interpreter
A. Koprowski and H. Binsztok (2010)

PEGs (parsing expression grammars) are very similar to *context-free grammars* (CFG) but

- The choice operator $e_1|e_2$ is prioritized: e_2 is only tried if e_1 fails;
- The repetition operators e^* and e^+ are greedy;
- (There are also predicates, $\&e$ and $!e$, which do not consume any input, and provide unlimited look-ahead and limited backtracking).

PEGs are unambiguous; they formalize what *recursive descent parsers* do (*packrat parsers* are recursive descent parsers with memoization).

Is there an optimal forecast combination?
C. Hsiao and S.K. Wan (2011)

There are many ways of combining forecasts f_{1t}, \dots, f_{nt} of a time series y_t into $\beta' f_t$.

With a small sample, the (unweighted) *simple average* \bar{y}_t works well. With more data, one can correct the bias $\mu + y_t$, or both bias and scale $\mu + c\bar{y}_t$.

Minimizing the mean square error

$$\begin{aligned} \phi(\beta) &= E[(\beta'x - y)^2] \\ &= E[\beta'x\beta'x] - 2E[\beta'xy] + E[y^2] \\ &= E[\beta'xx'\beta] - 2E[\beta'xy] + E[y^2] \\ &= \beta'E[xx']\beta - 2\beta'E[xy] + E[y^2] \end{aligned}$$

$$\phi(\beta + h) = \phi(\beta) + 2h'(E[xx']\beta - E[xy]) + O(\|h\|^2)$$

gives $\beta = E[xx']^{-1}E[xy]$. There are many variants:

- Linear regression;
- Linear regression with no intercept;
- Constrained linear regression, imposing $\beta' \mathbf{1} = 1$, *i.e.*, a convex combination of the forecasts;
- Inverse-mean-square combination (Bates-Granger), *i.e.*, using a diagonal estimator of $E[xx']$;
- Best subset regression, using the AIC, AICC or BIC to select the subset of forecasts to use;
- Bayesian model averaging, *i.e.*, $w_i \propto \exp -\frac{1}{2}\Delta\text{BIC}_i$, where ΔBIC_i is the difference between in BIC between model i and the best model.

One can try to replace the constraint $\beta' \mathbf{1} = 1$ with $\|\beta\|_2 = 1$. This gives the eigenvector e_i for the smallest eigenvalue λ_i , but it has to be rescaled to give an unbiased estimate: $e_i/e'_i \mathbf{1}$. The paper suggests to use the eigenvector minimizing $\lambda_i/(e'_i \mathbf{1})^2$ instead of the first. It is possible to improve the resulting estimator by estimating and correcting for the bias, and by discarding the worst performing forecasts (half of them, if the performance looks stable, or just a quarter), based on their variance or their mean square error.

Prediction with expert advice

N. Cesa-Bianchi and G. Lugosi (2006)

1. *Sequence prediction* tries to forecast the future values of a sequence without any underlying stochastic model: the sequence could be deterministic, stochastic, or even devised by an adversary, dynamically adapting to our behaviour.

Consider the following setup.

- N : number of experts
- n : number of observations
- i : expert
- t : time
- y_t : time series to forecast
- f_{it} : forecast of y_t by expert i
- \hat{p}_t : combination of the experts' forecasts
- $\ell(f_{it}, y_t)$: loss for expert i
- $\ell(\hat{p}_t, y_t)$: loss for the forecaster
- $h(f, y)$: payoff, $-\ell(f, y)$

The regret of not having followed expert i is

$$r_{it} = \ell(\hat{p}_t, y_t) - \ell(f_{it}, y_t)$$

$$R_{in} = \sum_{t=1}^n r_{it} \quad (\text{cumulated regret}).$$

We want to minimize the *cumulated regret* wrt the best expert,

$$\hat{L}_n = \min_{1 \leq i \leq N} L_{in}.$$

2. Here are a few weighted average predictors,

$$\hat{p}_t \propto \sum_{i=1}^N w_{i,t-1} f_{it}.$$

Polynomial	$w_{it} \propto (R_{it})_+^{p-1}$
Exponential	$w_{it} \propto \exp \eta R_{it} \propto \exp -\eta L_{it}$ $\eta = \sqrt{8 \log N/n}$
Gradient	$w_{it} \propto \exp -\eta \sum_{s < t} \nabla_{\hat{p}_s} \ell(\hat{p}_s, y_s) \cdot f_{is}$
Multilinear	$w_{it} \leftarrow (1 + \eta \cdot h(f_{it}, y_t)) \cdot w_{i,t-1}$

The polynomial and exponential predictors can be written $w_{it} \propto \nabla \Phi(R_{it})_i$, for some potential $\Phi(u) = \psi(\sum \phi(u_i))$ or, equivalently, $w_{it} \propto \phi'(R_{i,t-1})$. (While ψ does not affect the weights, different choices lead to different inequalities.)

The multilinear predictor increases or decreases the weights depending on the sign of the payoff.

Methods for pastcasting, nowcasting and forecasting using Factor-MIDAS, with an application to Korean GDP **H.H. Kim and N.R. Swanson (2016)**

Given two (or more) time series with different frequencies, Y (slow) and X (fast), the **MIDAS** model is a regression

$$Y_{y+k} = \beta_0 + \sum_{i=0}^{\ell} \beta_i X_{t-i} + \text{noise}$$

where the β_i have a sparse parametrization, *e.g.*, the *exponential Almon lag model* $\beta_i \propto \exp[\theta_1 i + \theta_2 i^2]$.

Given a large number of time series, “diffusion indices” are latent factors. To estimate them (by PCA), the time series have to be aligned, *e.g.*, by AR interpolation – other methods are available (and preferable): EM, state space model, etc.

A *factor-MIDAS* model is a MIDAS model with those latent factors as predictors.

The many revisions the data undergoes (the preliminary GDP is revised several times, often drastically) complicate the picture – each value has two timestamps: when it was published, and the period it refers to.

Deep symbolic representation learning for heterogeneous time series classification **S. Zhang et al.**

To classify heterogeneous (continuous and categorical) time series, discretize them, consider the values at a point in time as a word, and learn (end-to-end) word embeddings – or character embeddings, representing words as the direct sums of their characters.

PSF: introduction to R package for pattern sequence based forecasting algorithm **N. Bokde et al.**

The pattern sequence-based forecasting (PSF) algorithm forecasts time series by clustering (k -means) historical data into “patterns”. Only use if patterns are actually present (weather, energy, etc.).

***Interpreting finite automata
for sequential data***
C.A. Hammerschmidt et al. (2016)

Finite automata can model sequence data (model each word with an automaton; progressively merge them, selecting the “best” merge each time; stop when no good merge can be found); even large ones may be interpretable (use some graph layout algorithm and domain knowledge to interpret the clusters, bridges, etc.).

***Comparison of pattern detection methods
in microarray time series
of the segmentation clock***
M.L. Dequéant et al. (2008)

To detect patterns: uniformize the data and compute

- $\sigma/(\sigma_1 + \dots + \sigma_4)$, where σ_i is the standard deviation of the i th quarter of the data;
- The number (or proportion) of permutations with the same total variation;
- The persistence diagram of the sublevel sets $[x \leq a]$, for $0 \leq a \leq 1$.

***Detecting periodic patterns
in unevenly spaced gene expression time series
using Lomb-Scargle periodograms***
E.F. Glynn et al. (2005)

The **Lomb-Scargle periodogram** generalizes the discrete Fourier transform to unevenly spaced observations.

$$2\sigma^2 P(\omega) = \frac{\sum (y_i - \bar{y}) \cos^2(\omega(t_i - \tau))}{\sum \cos^2(\omega(t_i - \tau))} + \frac{\sum (y_i - \bar{y}) \sin^2(\omega(t_i - \tau))}{\sum \sin^2(\omega(t_i - \tau))}$$

$$\tau = \frac{1}{2\omega} \arctan \frac{\sum \sin 2\omega t_i}{\sum \cos 2\omega t_i}$$

$$H_0 : P(\omega) \sim \text{Exp}$$

***The cyclohedron test for finding periodic genes
in time course expression studies***
J. Morton et al. (2007)

The *cyclohedron test* tests if a time series (v_1, \dots, v_n) looks cyclic by computing its *signature* $\{\sigma_1, \dots, \sigma_{n-1}\}$

$$\sigma_1 = \{\text{index of the largest element}\}$$

$$\sigma_i = \{\text{index of the } i\text{th largest element}\} \cup \sigma_{k_{\text{left}}} \cup \sigma_{k_{\text{right}}}$$

where $\sigma_{k_{\text{left}}}$ is the latest σ_k containing $\delta_i - 1 \pmod{n}$. The test statistic is the *permutation count*, the number of permutations with the same signature.

***Factor models for matrix-valued
high-dimensional time series***
D. Wang et al. (2016)

Time series of matrices X_t (e.g., country \times variable \times time) can be described by a factor model $X_t = RF_tC' +$

E_t (it is not identifiable, but the column spaces of R and C are).

***On the relation between Gaussian process
quadratures and sigma-point methods***
S. Särkkä et al.

Sigma-point quadrature approximates integrals as

$$\int g(x)p(x)dx \approx w_i g(x_i),$$

where the w_i are predefined (from p) and the x_i 's are chosen according to some criterion, e.g., giving exact results on low-degree polynomials or minimizing the error on some class of functions. They are used in non-linear Kalman filters and can be interpreted as Gaussian process (GP) quadratures.

Lattice methods for multiple integration
I.H. Sloan and S. Joe (1994)

The *Fibonacci rule* estimates a 2-dimensional integral by evaluating it on a lattice

$$\iint_{[0,1]^2} f(x,y) dx dy \approx \frac{1}{F_k} \sum_{j=0}^{F_k-1} f\left(\left\{\frac{j}{F_k}(1, F_{k-1})\right\}\right)$$

where F_n is the n th Fibonacci number and $\{x\}$ is the fractional part of x . In dimension 2, they are the lattices with the best convergence rate.

***Monte Carlo
with determinantal point processes***
R. Bardenet and A. Hardy (2016)

Determinantal point processes (DPP) can be used to build repulsive particle systems, and give an alternative to quasi-Monte-Carlo integration or *scrambled sets* (randomized QMC). A *point process* is a random variable S whose values are finite subsets $[-1, 1]^d$. It may have an *n-correlation function* ρ_n , such that for all bounded measurable $\phi : [-1, 1]^d \rightarrow \mathbf{R}$,

$$E \left[\sum_{\substack{x_1, \dots, x_n \in S \\ \text{all different}}} \phi(x_1, \dots, x_n) \right] = \int_{[-1, 1]^d} \phi(x_1, \dots, x_n) \rho_n(x_1, \dots, x_n) \mu(dx_1) \cdots \mu(dx_n).$$

It is *determinantal* if the n -correlation functions exist and are given by a kernel,

$$\rho_n(x_1, \dots, x_n) = \det k(x_i, x_j)_{1 \leq i, j \leq n}.$$

For the orthogonal polynomial (OP) ensemble, the kernel is given by

$$k_N(x, y) = \sum_{k=0}^{N-1} \phi_k(x) \phi_k(y),$$

where the ϕ_k are the Gram-Schmidt orthonormalization of $x_1^{\alpha_1} \dots x_d^{\alpha_d}$, where the monomials are in the sup-graded lexicographic order.

$$\int f d\mu = E \left[\sum_{i=1}^N \frac{f(x_i)}{k_N(x_i, x_i)} \right]$$

(However, as N increases, the ϕ_k become difficult to compute and sampling from the DPP time-consuming.)

Why interval arithmetic is so useful
Y. Hijazi et al.

Affine arithmetic is a generalization of interval arithmetic.

$$\begin{array}{ccc} a \leq x \leq b & \text{vs} & a \leq \alpha x + \beta y \leq b \\ c \leq y \leq d & & c \leq \gamma x + \delta y \leq d \end{array}$$

Applications in computational geometry include computing surface intersections (ensuring that no bit of the intersection is lost), ray tracing, implicitly-defined surfaces.

A comparison of three high-precision quadrature schemes
D.H. Bailey et al. (2005)

The author's ARPEC C++/F90 multiprecision library also contains an implementation of PSLQ and quadrature schemes, $\int_{-1}^1 f \approx \sum w_j f(x_j)$:

Gauss : $x_j = \text{Roots of Legendre polynomials}$

$$w_j = \frac{-2}{(n+1)P'_n(x_j)P_{n+1}(x_j)}$$

Erf : $x_j = \text{erf}(hj)$

$$w_j = \frac{2}{\sqrt{\pi}} e^{-h^2 j^2}$$

Tanh-Sinh : $x_j = \tanh u_2$

$$w_j = \frac{u_1}{\cosh^2 u_2}$$

$$u_1 = \frac{\pi}{2} \cosh(hj)$$

$$u_2 = \frac{\pi}{2} \sinh(hj)$$

Dynamic trading with predictable returns and transaction costs
N. Gârleanu and L.H. Pedersen (2013)

In presence of transaction costs:

- Do not rebalance all the way to the target portfolio;
- The target portfolio is not the current optimal portfolio, but that in a few periods: the weights of all assets are shrunk, with that of those present because of fast signals shrinking faster.

The dynamic optimization problem can be solved exactly (Bellman equation).

Weighted elastic net penalized mean-variance portfolio design and computation
M. Ho et al. (2015)

Add an elastic net penalty to portfolio optimization problems to control overfitting and portfolio size.

Portfolio optimization based on stochastic dominance and empirical likelihood
T. Post and S. Karabati (2017)

The **empirical likelihood** (EL) tweaks the empirical distribution (from uniform on the sample to multinomial) to ensure some known properties of the population not satisfied by the sample (moment conditions) or add some prior information.

$$\begin{array}{ll} \text{Find} & \pi_1, \dots, \pi_n \\ \text{To maximize} & \sum \log \pi_i \quad (\text{MaxEnt}) \\ \text{Such that} & \sum \pi_i = 1 \\ & \forall j \sum_i \pi_i g_j(x_i) = 0 \quad (\text{Moments}) \end{array}$$

The moment conditions can also be inequalities; the maximum entropy objective is the multinomial log-likelihood. It is often used in econometrics, as an alternative to the generalized method of moments, when the moment conditions depend on a parameter θ , to be estimated with the π_i 's.

To account for dynamic patterns (ARMA, GARCH), blockwise empirical likelihood (BEL) uses overlapping data blocks instead of individual observations.

The resulting distribution can be used for second-order stochastic dominance portfolio optimization (a big linear problem). For instance, one can build a sector rotation strategy, maximizing the expected returns, using the 12-month momentum as an estimate of future returns, and moment conditions truncating the Fama-French-Cahart factors to their long-term 25%–75% quantiles.

All that glitters is not gold: comparing backtest and out-of-sample performance on a large cohort of trading algorithms
T. Wiecki et al.

The in-sample information ratio (IR) of a maximim IR strategy is not a good predictor of the out-of-sample IR, but other measures of performance fare better (volatility, maximum drawdown).

Instead of using the in-sample IR as an estimate, let machine learning build one, using a long list of performance and risk measures as features.

Dynamic quantile models of rational behavior
L. de Castro and A.F. Galvao (2017)

Instead of *expected* utility, maximize *quantile* utility.

The capacity of trading strategies

A. Landier et al. (2015)

Using the Garleanu-Pedersen model, *i.e.*, an AR(1) investment signal with persistence parameter ϕ ,

$$\begin{aligned}\Delta s_{t+1} &= -\phi s_t + \varepsilon_{t+1} \\ s_{t+1} &= (1 - \phi)s_t + \varepsilon_{t+1}\end{aligned}$$

with quadratic transaction costs $\frac{1}{2}\lambda(\Delta w)'\Sigma(\Delta w)$ and trading rate τ

$$x_t = (1 - \tau)x_t + \tau x_t^*$$

and with a few approximations, one can compute the relation between portfolio size and Sharpe ratio

$$\text{Size} = \frac{\text{SR}}{\lambda\phi^2} \left[\left(\frac{\text{SR}^*}{\text{SR}} \right)^{2/3} - 1 \right]^2.$$

For instance, if you are willing to accept a 30% drop in the Sharpe ratio, *i.e.*, $\text{SR} = 0.7 \times \text{SR}^*$, $\text{Size} = \text{SR}^*/10\lambda\phi^2$.

A Sharpe ratio neutral prior for Bayesian portfolio selection

R. Croessmann

In Bayesian portfolio selection,

$$\text{Maximize}_w \mathbb{E}_{X \sim N(\mu, \Sigma)} [U(w'X)],$$

$(\mu, \Sigma) \sim \text{Prior}$

the uninformative (improper) Jeffrey prior $p(\mu, \Sigma) \propto |\Sigma|^{-(N+1)/2}$ does not account for known information:

- Very high Sharpe ratios are unlikely, while Jeffrey's prior gives

$$\begin{aligned}\text{SR}_{\max} &= \sqrt{\mu'\Sigma^{-1}\mu} \\ p(\text{SR}_{\max}) &\propto \text{SR}_{\max}^{N-1}\end{aligned}$$

- High returns are a compensation for higher risks, while the prior assumes $\mu \perp \Sigma$.

Instead, use

$$\begin{aligned}X &\sim N(\beta\lambda, \beta\beta' + \Sigma) \\ \Sigma &= \text{diag}(\tau) \Omega \text{diag}(\tau) \\ p[\Omega] &\propto |\Omega|^{N(N-1)/2} \prod_i |\Omega_{-i, -i}|^{-(N+1)/2} \\ \beta, \lambda, \tau &\text{ flat;}\end{aligned}$$

this gives uniform marginal posteriors for $\text{Cor}(X_i, X_j)$.

Optimal selection of large portfolios: aggregation is better than ignoring the return constraint

G.Y. Ban and C. Chen

The out-of-sample Sharpe ratio deteriorates as the number of assets increases. Instead of optimizing a portfolio with all assets, group them, randomly, into

“super-assets”; replace each group with the equal-weighted portfolio of its constituents; optimize.

Instead of randomly grouping the assets, group them in pairs with high correlation (or low covariance) and put the elements of each pair in separate super-assets.

Optimizing optimal portfolio choice

Y. Jin (2017)

Instead of looking for better estimates of Σ , Σ^{-1} , or μ , one can directly look for better estimates of the optimal (unconstrained) weights, $\Sigma^{-1}\mu$. Since they can be defined by an optimization problem, $\text{Argmin}_w \|\Sigma w - \mu\|_2^2$, one can add an L^1 penalty to get a sparse portfolio,

$$\hat{w} = \text{Argmin}_w \frac{1}{2} \left\| \hat{\Sigma} w - \hat{\mu} \right\|_2^2 + \lambda \|w\|_1.$$

Under reasonable assumptions (e.g., a shrinkage estimator for $\hat{\Sigma}$, $\|\hat{w} - w^*\| = O_p(\sqrt{S \log(N)/T})$, where $S = \text{Min}\{T, N\}$ and $O_p(\cdot)$ means $O(\cdot)$ with high probability – the best possible bound is $O(\sqrt{S \log(N)/T})$.

Bayesian emulation for multi-step portfolio decisions

K. Irie and M. West (2016)

Optimization problems can often be recast as estimation problems, by interpreting the loss function as a log-likelihood and the penalties as priors. For instance, multi-period optimization problems become state space models, which can be estimated with a “forward filtering backward smoothing” (FFBS) algorithm (e.g., Kalman filter, Viterbi algorithm).

Portfolio construction by mitigating error amplification: the bounded-noise portfolio

L. Zhao et al.

Eigenvectors can only be reliably estimated if their eigenvalues are isolated (*eigengap*): this is only the case for large eigenvalues.

The minimum variance portfolio is a linear combination of that for large eigenvalues (“signal”) and that for small ones (“noise”), $w = \alpha w_S + (1 - \alpha)w_N$. The noise portfolio can be replaced with a robust alternative, minimizing an upper bound on volatility. The resulting optimization problem has the usual form, with the variance Σ replaced with $\Sigma + mNN'$, where the columns of N are the noise eigenvectors (the number of noise eigenvectors is estimated by bootstrap).

Dynamical system theory of periodically collapsing bubbles

V.I. Yukalov et al.

Interactions between a stock and a bond can be modeled as

$$\begin{aligned}\dot{x} &= x - x^2 e^{-bxz} \\ \dot{z} &= z - z^2 e^{-ax},\end{aligned}$$

with $a \in [-.5, .5]$, $b \in [0, 1]$. The presence of bifurcations explains periodically exploding bubbles.

Measuring uncertainty
K. Jurado et al. (2015)

Measure the uncertainty of a macroeconomic series as the standard deviation of the forecasted error,

$$Y_t \sim \sum_i X_{it} \quad \text{factors}$$

$$X_{it} \sim \text{Time series model} \quad \text{factor evolution}$$

and use a stochastic volatility model for the innovations in both models. Finally, aggregate (weighted average) the uncertainties of several series.

Multivariate geometric expectiles
K. Herrmann et al. (2017)

Univariate quantiles can be defined from an asymmetric absolute value loss; they generalize the median.

$$\rho_\alpha(t) = \begin{cases} \alpha t & \text{if } t \geq 0 \\ (1 - \alpha)t & \text{if } t \leq 0 \end{cases}$$

$$F^{-1}(\alpha) = \underset{c \in \mathbf{R}}{\text{Argmin}} E\rho_\alpha(X - c).$$

The univariate **expectiles** are defined from an asymmetric square loss; they generalize the mean.

$$\lambda_\alpha(t) = \begin{cases} \alpha t^2 & \text{if } t \geq 0 \\ (1 - \alpha)t^2 & \text{if } t \leq 0 \end{cases}$$

$$E(\alpha) = \underset{c \in \mathbf{R}}{\text{Argmin}} E\lambda_\alpha(X - c).$$

In higher dimensions (implementation in **qrmtools**):

$$\Phi_u(t) = \|t\|_2 + \langle u, t \rangle \quad u \in B(0, 1)$$

$$\text{VaR}_\alpha X = \underset{c \in \mathbf{R}^d}{\text{Argmin}} E\Phi_\alpha(X - c)$$

$$\Lambda_u(t) = \|t\|_2 (\|t\|_2 + \langle u, t \rangle)$$

$$e_\alpha(X) = \underset{c \in \mathbf{R}^d}{\text{Argmin}} E\Lambda_\alpha(X - c).$$

A generalized contagion process with an application to credit risk
A. Dassios and H. Zhao (2016)

A **Cox process** (doubly stochastic model) is a Poisson process in which the intensity $\lambda(t)$ is itself stochastic. A *dynamic contagion process with diffusion* (DCPD) is a Hawkes process $(T_i)_{i \geq 0}$ with both self-excited and external jumps, with noise.

$$\lambda(t) = a \quad \text{mean-reverting level}$$

$$+ \sum_{T_i < t} Y_i e^{-\delta(t-T_i)} \quad \text{internal jumps}$$

$$+ \sum_{S_i < t} X_i e^{-\delta(t-S_i)} \quad \text{external jumps}$$

$$+ \sigma \int_0^t e^{-\delta(t-s)} \sqrt{\lambda(s)} dW_s \quad \text{noise}$$

Data-driven partition-of-unity copulas with applications to risk management
D. Pfeifer et al. (2017)

The following procedure defines a bivariate density:

- Take two 1-parameter families ϕ, ψ of (discrete) distributions on \mathbf{N} , indexed by $[0, 1]$, e.g., negative binomial or Poisson;
 - Take a discrete distribution on $\mathbf{N} \times \mathbf{N}$ whose marginals are the averages of ϕ and ψ :
- $$p_{i,\cdot} = \int_0^1 \phi_i(u) du \quad p_{\cdot,j} = \int_0^1 \psi_j(v) dv;$$
- Set $c(u, v) = \sum_{ij} p_{ij} \frac{\phi_i(u)}{p_{i,\cdot}} \frac{\psi_j(v)}{p_{\cdot,j}}$.

Robust return risk measures
F. Bellini et al. (2016)

The *Orlicz premium* of a positive random variable X is the only solution $H_\Phi(X)$ of

$$E \left[\Phi \left(\frac{X}{H_\Phi(X)} \right) \right] = 1,$$

where $\Phi : [0, +\infty) \rightarrow [0, +\infty)$ is convex, $\Phi(0) = 0$, $\Phi(1) = 1$, $\Phi(\infty) = \infty$.

The utility-based *shortfall risk* of a real random variable Y is $\rho_\ell(Y) = \inf\{m : E[\ell(Y - m)] \leq 0\}$ where $\ell : \mathbf{R} \rightarrow \mathbf{R}$ is non-decreasing and $\ell(-\infty) < 0 < \ell(+\infty)$. These notions are related:

$$\Phi(x) = 1 + \ell(\log x)$$

$$H_\Phi(X) = \exp(\rho_\ell(\log X))$$

$$\rho_\ell(Y) = \log(H_\Phi(\exp Y)).$$

They can be robustified, e.g., by replacing $E[\cdot]$ with $\sup_{Q \in \mathcal{S}} E_Q[\cdot]$ (one can also consider a set of Young functions Φ).

The *Hazendonck-Goovaert risk measure* is

$$\Pi_\Phi(X) = \inf_{x \in \mathbf{R}} x + H_\Phi((X - x)_+).$$

Multifactor risk models and heterotic CAPM
Z. Kakushadze and W. Yu (2016)

Build your own risk model:

- Separately model variances and correlations;
- Use a factor model for the correlation;
- Use (6-digit) industries as factors, but estimate the model in a russian-dolls fashion.

Multiportfolio time consistency for set-valued convex and coherent risk measures
Z. Feinstein and B. Rudloff

Intuitively, a *set-valued risk measure* is a collection of portfolios covering the risk of the asset of interest. A risk measure is *time-consistent* if a portfolio covering the risk at time t also covers the risk at any earlier time $s < t$.

***Latent common return volatility factors:
capturing elusive predictive accuracy gains
when forecasting volatility***
M. Cheng et al. (2017)

Stock volatility forecasts are usually computed independently for each stock, ignoring dependencies. Instead:

- Compute the realized volatility (RV: multipower, multiscale, etc.) of each stock;
- Use the lasso to find stocks whose RV is close to that of the target;
- Compute a sparse PCA on them, to extract “volatility factors” (or just use a factor model);
- Fit a heteroskedastic autoregressive (HAR) model $y_{[t,t+1]} \sim y_{[t-1,t]} + y_{[t-4,t]} + y_{[t-21,t]}$ to the RV of the stock of interest, with the factors as exogenous variables.

***Introduction to noise-reduced correlations
using singular spectrum analysis***
J.W. Dash et al. (2017)

The signal-to-noise ratio of a correlation matrix can be defined as

$$\text{SNR} = \frac{\sum_{i < i_c} \lambda_i}{\sum_{i \geq i_c} \lambda_i}$$

where the cutoff i_c is determined by finite (not asymptotic) random matrix theory (RMT).

To estimate a correlation matrix, start with log-prices, normalize them, smooth them with singular spectrum analysis (SSA), compute the returns, and then the correlations.

***The 7 reasons
most machine learning funds fail***
M. López de Prado (2017)

Do not compute returns by differentiating prices: *fractional differentiation* ($\alpha = .5$) is sufficient to yield a stationary process.

Do not use clock time but market time (subordinated process using, not time, but volume or dollar bars).

Do not forecast “up” or “down” for a fixed horizon, but use *three barriers*: up and down thresholds, and a time limit; the labels to forecast may overlap: weigh them accordingly.

To avoid overfitting, make sure there is no overlap between training and test sets, and use the *deflated Sharpe ratio*.

***A tug of war:
overnight versus intraday expected returns***
D. Lou et al. (2014)

Some investment signals realize their profits overnight (momentum, reversal), some intraday (the rest).

101 formulaic alphas
Z. Kakushadze (2015)

101 technical indicators.

***Extracting consumer demand:
credit card spending and post-earnings returns***
S. Agarwal et al. (2017)

Consumer spending (credit card data, email invoice data (Slice)) and spending surprises predict future earnings, sales surprises and future returns

***What makes stock prices move?
Fundamentals vs investor recognition***
S. Richardson et al. (2012)

Stocks with low investor recognition (number of institutional shareholders) have a lower price and higher expected returns; realized returns as higher when they gain more recognition.

Facts about formulaic value investing
U.W. Kok et al. (2017)

Formulaic value investing (P/E and other financial ratios) only highlights companies with temporarily high inflated earnings.

A framework for value investing
S. Chee et al. (2013)

The discounted dividend model computes an intrinsic value from dividend forecasts and an “appropriate discount rate”. Instead, the Ohlson model estimates this discount rate (prospective yield) so that the value equals the current market price,

$$P = \sum_{t \geq 1} \frac{E[\text{Div}_t]}{(1+y)^t}.$$

Estimating the dividends as $\text{Div}_t = \text{Earnings}_t - \Delta \text{BV}_t$,

$$y \approx \left[\frac{E[Z_T]}{P} + 1 \right]^{1/T} - 1$$

where Z_T is the expected aggregate cum-dividend earnings for the next T periods.

Accounting anomalies, risk and return
S.H. Penman and J. Zhu (2011)

Cash accounting recognizes revenue when it is actually received.

Accrual accounting recognizes revenue when it becomes certain.

One could go one step further and recognize revenue when it can be forecasted: this explains many accounting “anomalies” – higher returns are a compensation for the uncertainty of future profits.

***A dynamic model of characteristic-based
return predictability***
A. Alti and S. Titman (2017)

Model firms as a combination of assets and projects, each project yielding a deterministic cash flow from inception to termination and liquidation cash flows (salvage value), with younger firms having more, and more profitable projects.

$$\begin{aligned} z_i &\in \{\text{early, mature, dying, dead}\} && \text{state} \\ dk_{it} &= k_{z_i}(1 - \lambda)(dt + dM_t) && \text{capital} \\ df_{it} &= (a_{z_i}k_{z_i} - \lambda f_{it})(dt + dM_t) && \text{profitability} \\ dM_t &= \mu_t dt + \sigma_M d\omega_t^M && \text{systematic disruption} \\ d\mu_t &= -\rho\mu_t dt + \sigma d\omega_t^\mu && \text{disruption climate (hidden)} \\ ds_t &= \eta d\omega_t^M + \sqrt{1 - \eta^2} d\omega_t^s && \text{soft interaction signal} \end{aligned}$$

Investors' overconfidence on the precision of their estimates creates mispricing.

Having a model of the economy (knowing the soft information s instead of just the performance of the sorted portfolios) helps (it doubles the information ratio).

***Structure in the Tweet haystack: uncovering
the link between text-based sentiment signals
and financial markets***
A. Groß-Klußmann et al. (2015)

Manually identify 200 "expert" Twitter users (from web pages: "top 50 Twitter influencers", "who to follow", etc.); measure tweet sentiment using the Harvard general inquirer and Loughran-McDonald lists (in the latter, adapted to finance, "tax" is no longer negative), without stemming (this can mix up some positive and negative words). Sentiment and sentiment disagreement have some predictive power on market returns and volatility.

***Stock return predictability:
consider your open options***
F. Farazmand and A. de Souza (2017)

The put-call open interest ratio (aggregated put open interest, divided by aggregated call open interest, for liquid options, *i.e.*, $K/S \in [.8, 1.2]$ and maturity $\in [30d, 90d]$), for high-volatility (arbitrage-constrained) stocks is a good predictor of future market returns. This can be explained by investors' limited attention.

Beta dispersion and market timing
L.C. Kuntz (2016)

After an exogenous shock to the market, stocks with a high beta are in trouble; if there are many of them, *i.e.*, in markets with high beta dispersion (measured by the 10%-90% interquantile range), this could trigger a second, endogenous shock.

***Investor attention and sentiment:
risk or anomaly?***
M.C. Bucher (2017)

Google searches to measure investor attention (index, stock tickers) and sentiment (panic words, Fears index).

***Algorithmic differentiation in finance
explained***
M. Henrard (2017)

R in Finance 2017

1. GAS (generalized autoregressive score) models generalize GARCH models (a Gaussian GAS is a GARCH) but can be made less sensitive to outliers, leading to better VaR estimation.

Markov switching GARCH models (implemented in MSGARCH) give better VaR forecasts than single-regime ones. Prefer GJR, with skewed Student innovations; prefer MCMC to MLE to account for parameter uncertainty.

Continuous GARCH models (**COGARCH**) generalize GARCH models to non-regularly-spaced observations. In high dimensions, they can be combined with ICA (independent component analysis).

The yuima package estimates and simulates SDEs; it now has an HTML GUI.

Two time series X, Y are **partially cointegrated** if some linear combination $Y - \beta X$ is partially autoregressive, *i.e.*, the sum of a random walk and an AR process.

For many multivariate diffusions, an arbitrarily close approximation (in terms of Hermite polynomials) of the transition function is known in closed form: the MLE is easy to compute. (If not all components are observed, further approximations are available.) The MLEMVD package implements this for many for many 1- or 2-dimensional diffusions

$$\begin{aligned} dX_t &= \mu(X_t)dt + \Sigma(X_t)dW_t \\ \mu : \mathbf{R}^m &\rightarrow \mathbf{R}^n \\ \Sigma : \mathbf{R}^n &\rightarrow \mathbf{R}^{n \times n} \end{aligned}$$

2. The standard error of performance measures T (Sharpe, Sortino, etc.) can be estimated from the sample cumulative distribution function F_n by bootstrap (or **block bootstrap**, in presence of serial correlation) or with **influence functions**

$$\text{IF}(r, T, F_n) = \frac{d}{d\gamma} T((1 - \gamma)F_n + \gamma\delta_r) \Big|_{\gamma=0}$$

(for many performance measures, they can be explicitly computed) since the asymptotic variance is

$$\text{Var}[T(F_n)] \underset{n \rightarrow \infty}{\sim} E[\text{IF}(r_1, T, F)]^2.$$

With serial correlation,

$$\begin{aligned}\text{Var}[T(F_n)] &= \sum_{\ell \in \mathbf{Z}} E[\text{IF}(r_1, T, F_n) \text{IF}(r(1 + \ell), T, F_n)] \\ &= \sum_{\ell \in \mathbf{Z}} C(\ell) \quad \text{autocovariance density} \\ &= S(0) \quad \text{spectral density}\end{aligned}$$

and $S(0)$ can be estimated from a polynomial fit to the periodogram of the IF time series $\text{IF}(r_t, T, F_n)$.

Whenever possible, use the *two-scale estimator* (TSRV) of volatility.

The (block) rearrangement algorithm (BRA) takes a matrix and reorders the elements within each column to minimize the variance of row sums. It can be used to estimate the worst-case portfolio VaR when only the asset return distributions are known (and not their copula). When applied to $(X_1, \dots, X_n, -S)$, where $S = X_1 + \dots + X_n$, it gives a dependence structure close to the maximum entropy one (the correlation matrix of the reordered (X_1, \dots, X_n) has maximum determinant), *i.e.*, answers the question: “if we know the marginal distribution of X_1, \dots, X_n and S , what is the most likely dependence structure?”

When estimating sample comoment tensors, do not re-compute entries already computed (the tensors are symmetric), and use shrinkage, towards a factor model

$$\begin{aligned}X &= BF + \varepsilon \\ \Sigma &= B\Sigma_F B' + \Delta \\ \Phi &= B\Phi_F(B \otimes B)' + \Omega \\ \Psi &= B\Psi_F(B \otimes B \otimes B)' + \Gamma.\end{aligned}$$

Covariance matrix estimators can be more or less structured:

$$\begin{aligned}(1) \quad R_t &\sim N(0, \Sigma_t) & (3) \quad R_t &= F_t \beta'_t + \varepsilon_t \\ & & \beta_t &= \beta_{t-1} + \eta_t \\ (2) \quad R_t &= F_t \beta'_t + \varepsilon_t & \varepsilon_t &\sim N(0, \Sigma_t) \\ \varepsilon_t &\sim N(0, \Sigma_t) & \varepsilon_t &\sim N(0, \Gamma_t)\end{aligned}$$

3. Do not just prefer the “best” strategy: also look at its *sensitivity* to parameter estimation. This is the usual bias-variance tradeoff: you may prefer a slightly worse performance if it is more certain.

It is difficult to select a good manager, especially if there are many of them and/or there is not enough history. Prefer *ensemble methods*: instead of selecting a single manager, select several.

The following stochastic, continuous-time model of **private equity** cash flow and value, which can accommodate fees, should replace the Yale model (deter-

ministic, with no fees).

$$\begin{aligned}dV &= V(\mu dt + \beta \sigma dB + \sigma dB) + dD - dR & \text{Value} \\ dD &= \delta(I - D)dt & \text{Cumulative drawdowns} \\ dR &= vV dt & \text{Cumulative distributions} \\ \delta &= \delta_0 + \sigma B & \text{Drawdown rate} \\ v &= v_0 t + \sigma B & \text{Distribution rate} \\ B & & \text{Different Brownian motions} \\ \sigma & & \text{Different volatilities} \\ dC &= Cr dt - dD + R dt & \text{Cash flow}\end{aligned}$$

This leads to several measures of risk:

– Market: $\text{VaR}[\text{P\&L}]$, where

$$\text{P\&L}_{[t, t+h]} = d_{[t, t+h]} P_{t+h} - P_t,$$

d is the discount factor and $P = V + C$;

– Liquidity: VaR of the liquidity-adjusted P&L,

$$(1 - \pi_{t+h})V_{t+h} + C_{t+h} - P_t,$$

where $d\pi = \kappa(\theta - \pi) + \sigma dB$ is the secondary market discount rate;

– Cash flow: $\text{VaR}[C_{t+h} - C_t]$.

4. For a sparse solution to your portfolio optimization problems, add an L^1 constraint; this also improves performance and can be generalized to structured sparsity.

Risk parity can be generalized to higher moments. The **mrpc** package minimizes the discrepancy between the contribution to second-, third- or fourth-order risk or the assets, or some linear combination of those discrepancies.

$$\begin{aligned}v &= E(r_p - \mu_p)^2 \\ &= E[r_p - \mu_p \otimes (r_p - \mu_p)] \\ &= E\left[\sum w_i(r_i - \mu_i) \otimes (r_p - \mu_p)\right] \\ s &= E\left[\sum w_i(r_i - \mu_i) \otimes (r_p - \mu_p)^{\otimes 2}\right] \\ \kappa &= E\left[\sum w_i(r_i - \mu_i) \otimes (r_p - \mu_p)^{\otimes 3}\right]\end{aligned}$$

$$\frac{\partial v}{\partial w} = 2M_2 w, \quad \frac{\partial s}{\partial w} = 3M_3(w \otimes w), \quad \frac{\partial \kappa}{\partial w} = 4M_4(w \otimes w \otimes w)$$

$$M_i = E[(r - \mu)^{\otimes i}]$$

Find

w

To minimize $\lambda_1 \text{Var } w \otimes \frac{\partial v}{\partial w} + \lambda_2 \text{Var } w \otimes \frac{\partial s}{\partial w} +$

$$\lambda_3 \text{Var } w \otimes \frac{\partial \kappa}{\partial w}$$

Such that $w' \mathbf{1} = 1, \quad w \geq 0$

PortfolioAnalytics can optimize (small) portfolios and generate random portfolios.

5. The score difference, in a soccer match, can be modeled as a difference of Poisson processes (a **Skel-lam process**) and calibrated using the odds provided by the bookie (dynamically); one can also define an “implied volatility” $\sqrt{\lambda_t^A + \lambda_t^B} \sqrt{1 - t}$, $t \in [0, 1]$.

Ratings (credit ratings, Morning Star stars, etc.) are often estimated independently in different categories, on a curve (*i.e.*, not calibrated) and therefore not comparable. Yet, people compare them.

The overnight gap on ex-div date is different from the dividend, because of taxes.

The `obmodeling` package models the order book (VPIN, etc.).

Not all brokers are liquidity providers; high-frequency traders seem to be liquidity-takers.

6. The `sn` package provides a skew-T distribution.

The risk-neutral density can be modeled as a piecewise constant function, generating the correct call and put prices (minimizing the squared relative error).

7. `MXNet` is available from R; it also works on Windows and can leverage your GPU.

Email length, and perhaps also sentiment was a better predictor of future returns than new sentiment in the two years leading to Enron's demise. **Topic analysis** (LDA) may also be interesting.

The `mlr` package now provides the same interface for machine learning (regression, etc.) and time series forecasting (ARIMA, GARCH, etc.)

Several talks compared ML algorithms (Adaboost, MARS, Neural networks and error correcting models (ECM) are very similar – prefer Adaboost and MARS).

Projection methods turn big data into data. A random orthogonal projection on a 1-dimensional subspace gives a lower bound for pairwise distances; **random projections** can be used to find the most correlated columns in a large matrix (`tcors`). Sparse SVD (`irlba`) can help compute the most central nodes of a large graph or bicluster large matrices (`s4vd`).

8. If the prior is not a single distribution but a family of distribution, then so is the posterior: it is a *regularization path*, and can be used to choose the “best” prior and also for prior sensitivity analysis.

To model the time evolution of multivariate yield curves $y_{\text{market},t}$ (tenor), write them as linear combinations of a few “principal” yield curves, modeled with splines, with weights changing as random walks. This is implemented in the `FDLM` package [but this could probably be implemented in Stan, and HMC should be better than Gibbs sampling].

9. Microsoft R server lets you expose your code as an API (not unlike OpenCPU). You can also put your R code in a stored procedure (as with Postgres).

The `roll` package provides a few (fast, parallelized with `RcppParallel`) rolling statistics: regression, PCA, correlation matrices, etc.

`ztsdb` is a time series database (C++, either file-based or client-server), append-optimized, with an R interface, and an R-like language.

Syberia is a workflow framework for data science in R.

The `rTRNG` package provides random numbers for parallel MCMC computations

You do not have big data. If you want speed, try `VowpalWabbit` and `xgboost`. If you really want something distributed, try `H2O` (it often turns out to be faster than `Spark`).

10. Some of the presentations showcased analyses with `plotly`, `shiny`, `flexdashboard`, `dygraph`, `rnaturalearth`, etc.

The tidyverse assumes data is in dataframes, while financial data tends to be in xts objects. The `tidyquant` package provides a “tidy” interface to manipulate data in xts objects.

Five stages of accepting constructive mathematics A. Brauer (2016)

1. **Constructive mathematics** is mathematics without the law of the excluded middle (and, therefore, without the axiom of choice):

- There are still proofs by negation ($P \Rightarrow \perp$ is the definition of $\neg P$), but no proofs by contradiction (proving $\neg P \Rightarrow \perp$ only gives $\neg\neg P$, not $\neg P$).
- The notions of non-empty set ($A \neq \emptyset$) and inhabited set ($\exists x \in A$) are different.

2. Since we have removed an axiom, we can add more, which sometimes leads to surprising consequences, e.g.,

- All functions are continuous;
- Subsets of finite sets need not be finite (*i.e.*, in bijection with some $\llbracket 1, n \rrbracket$);
- There is an injection $\mathbf{N}^{\mathbf{N}} \hookrightarrow \mathbf{N}$;
- There is an unbounded continuous map $[0, 1] \rightarrow \mathbf{R}$;
- \mathbf{R} has zero measure;
- There is an unbounded, increasing sequence without an accumulation point.

3. Models of constructive mathematics include:

- Classical mathematics;
- Computer programs (realizability);
- Sheaves on a topological space X (the truth values are the open sets of X), a topos;
- *Locales* (partially-ordered sets with properties similar to those of the open sets of a topological space; points are not needed: the regular open subsets of \mathbf{R} , *i.e.*, subsets U such that $U = \overset{\circ}{\bar{U}}$, is a sublocale of \mathbf{R} with no points).

4. Some results need to be adapted (e.g., the intermediate value theorem remains true with altered conclusions or additional assumptions, e.g., an approximate solution, a monotonic function, or isolated roots) and new objects can be defined: for instance, the set of random reals of \mathbf{R} minus all subsets of measure zero – it has full measure, but no points.

*What is the best fractional derivative
to fit data?*
R. Almeida

Phenomena traditionally modeled by differential equations such as

$$y' = k, \quad y' = ky, \quad y' = k(y - a), \quad y' = f(t)$$

may be more accurately described by fractional differential equations

$$D^\alpha y = k, \quad D^\alpha y = ky, \quad D^\alpha y = k(y - a), \quad D^\alpha y = f(t)$$

for some $\alpha \in (0, 1)$. The fractional differential and integral are, for $\alpha \in (n - 1, n)$,

$$D^\alpha f(x) = \frac{1}{\Gamma(n - \alpha)} \int_a^x (x - t)^{n - \alpha - 1} f^{(n)}(t) dt$$

$$D^{\alpha, \psi} f(x) = \frac{1}{\Gamma(n - \alpha)} \times \int_a^x \psi'(t) (\psi(x) - \psi(t))^{n - \alpha - 1} \left(\frac{1}{\psi'(t)} \frac{d}{dt} \right)^n f(t) dt$$

$$I^{\alpha, \psi} f(x) = \frac{1}{\Gamma(\alpha)} \int_a^x \psi'(t) (\psi(x) - \psi(t))^{\alpha - 1} f(t) dt.$$

Bayesian inference of log determinants

J. Fitzsimons et al.

To compute log-determinants of large kernel matrices as a Bayesian inference problem, use

$$\log \det A = \text{tr} \log A$$

$$\log(I - A) = \sum_{k \geq 1} A^k$$

$$\text{tr} A = \mathbb{E}_{\mathbf{r} \sim N(0, 1)} \mathbf{r}' A \mathbf{r}$$

and use probabilistic numerics for the *stochastic trace estimation*.

Exploiting gradients and Hessians in Bayesian optimization and Bayesian quadrature

A. Wu et al.

In Bayesian optimization, if the gradient or the Hessian is readily available, use it: the joint process $(f, \nabla f)$ is still Gaussian, with covariance function

$$\begin{pmatrix} k(x, y) & \nabla_x k(x, y) \\ \nabla_y k(x, y) & \nabla_x \nabla_y k(x, y) \end{pmatrix}.$$

The algorithms do not change, they no not use directly the derivatives, but conditioning on this additional information helps.

The kernel becomes ill-conditioned much more quickly: if the bandwidth is small, rescaling can help; if it is large, try moving to the spectral domain:

$$f \sim \text{GP} \implies \mathcal{F}f \sim \text{GP}.$$

On the construction of probabilistic Newton-type algorithms

A.G. Wills and T.B. Schön

Model the Hessian as a Gaussian process, updated with 1-dimensional information at each step (quasi-Newton), and use this expectation for the Newton step in optimization problems.

No spurious local minima in nonconvex low rank problems: a unified geometric analysis

R. Ge et al.

Many nonconvex optimization problems (even asymmetric ones) have a well-behaved loss landscape: local minima are also global (and saddle points have at least one negative eigenvalue in their Hessian).

A probabilistic linear genetic programming with stochastic context-free grammar for solving symbolic regression problems

L.F. Dal Piccol Sotto and V.V. de Melo

Use a stochastic grammar as prior for symbolic regression.

Structural change in (economic) time series

C. Kleiber (2016)

To test for the presence of structural change

$$\begin{array}{ll} \text{Univariate signal} & y_t = \mu_t + \varepsilon_t \quad \mu_t = \text{constant?} \\ \text{Regression} & y_t = x_t' \beta_t + \varepsilon_t \quad \beta_t = \text{constant?} \end{array}$$

one can look at the cumulated residuals (**empirical fluctuation process**, EFP, Rec-CUMSUM)

$$\hat{\mu}_k = \frac{1}{k} \sum_1^k y_t$$

$$e_t = y_t - \hat{\mu}_{t-1}$$

$$S_T = \frac{1}{\hat{\sigma} \sqrt{T}} \sum e_t$$

which can be approximated by a Brownian motion, and look for excessive fluctuations (crossing certain boundaries).

Instead, one can look at the OLS-CUMSUM process

$$e_t = y_t - \hat{\mu}_T$$

$$S_k = \frac{1}{\hat{\sigma} \sqrt{T}} \sum_1^k e_t$$

which can be approximated by a Brownian bridge; fluctuations can be identified by looking at $\text{Max}_t |S_t|$.

The location of the breakpoints can be estimated by minimizing the sum of squared residuals, with dynamic programming; the BIC can help choose the number of breakpoints.

In R, check the **strucchange**, **ecp** and **wbs** packages.

A nonparametric approach for multiple changepoint analysis of multivariate data
D.S. Matteson and N.A. James (2013)

To detect a changepoint in a multivariate time series $X_{1:n}$, compute

$$\operatorname{Argmax}_k \frac{k(n-k)}{n} \mathcal{E}(X_{1:n}, X_{k+1:n})$$

where

$$\mathcal{E}(X, Y) = 2E|X - Y|^\alpha - E|X - X'|^\alpha - E|Y - Y'|^\alpha$$

is the distance correlation (which can be seen as a divergence between characteristic functions).

In presence of multiple changepoints, prefer

$$\operatorname{Argmax}_{k,\ell} \frac{k\ell}{k+\ell} \mathcal{E}(X_{1:k}, X_{k+1:k+\ell})$$

and proceed recursively. In R: `ecp::e.divisive`.

Use permutation tests to assess the significance of the breakpoints and decide when to stop.

The agglomerative variant of the algorithm, merging the adjacent clusters minimizing

$$\frac{|C_i| |C_{i+1}|}{|C_i| + |C_{i+1}|} \mathcal{E}(C_i, C_{i+1}),$$

is faster.

Wild binary segmentation for multiple change-point detection
P. Fryzlewicz (2014)

Binary segmentation splits a time series at the point maximizing the CUSUM statistic (equivalently, it fits a piecewise constant function with exactly one jump to the data), and proceeds recursively to find more changepoints. It does not work well in presence of several breakpoints.

Wild binary segmentation is similar, but works on randomly-selected subsequences, of various lengths. R implementation in `wbs`.

Optimal detection of changepoints with a linear computational cost
R. Killick et al. (2012)

Optimal partitioning uses dynamic programming to find the breakpoints minimizing some criterion, with a penalty β to control this number

$$\begin{aligned} F(s) &= \min_{\substack{\tau \text{ partition} \\ \text{of } 1:s}} \sum_i \operatorname{Cost}(y_{\tau_i+1:\tau_{i+1}}) + \beta \\ &= \min_{t < s} F(t) + \operatorname{Cost}(y_{t+1:s}) + \beta. \end{aligned}$$

The computations can be pruned (pruned exact linear time, PELT) using the fact that, under reasonable assumptions, if $F(t) + \operatorname{Cost}(y_{t+1:s}) + k \geq F(s)$, then t cannot be the last optimal changepoint for any $T > s$.

Local extremes, runs, strings and multiresolution
P.L. Davies and A. Kovac (2001)

After smoothing a time series, one expects the residuals to be similar to white noise; in particular, their signs should not present long *runs*.

To detect extrema in a noisy time series (to approximate a time series with a function with few extrema), one can try to minimize the number of extrema among approximations of the time series for which the maximum length of the runs is below some threshold. This can be generalized to a multiresolution condition, where 2^k consecutive residuals are aggregated, for all k .

Given upper and lower bounds on the unknown values of a time series observed with noise, the *taut string method* looks for the shortest curve between those bounds (it can be computed in linear time, see `fntnonpar::mintvnorm(..., method=1)`).

Dynamic Bayesian combination of multiple imperfect classifiers
E. Simpson et al. (2012)

Consider k judges inferring the labels of N objects:

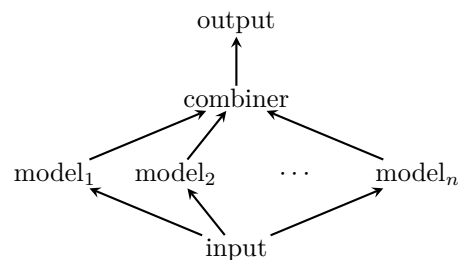
- κ : label distribution (multinomial)
- t_i : label of object i
- c_{ik} : label assigned to object i by judge k
- $\pi_{k\ell j} : P[\text{judge } k \text{ says } \ell \text{ when the true label is } j]$
- α : Dirichlet prior on π
- ν : Dirichlet prior on κ

The **IBCC** (independent Bayesian combination of classifiers) model can be estimated by MAP (maximum a posteriori), Gibbs sampling or, better, variational Bayes (VB).

Bayesian classifier combination
H.C. Kim and Z. Ghahramani (2012)

Contrary to IBCC, Bayesian model averaging (BMA) assumes the classifiers are somewhat correct (unbiased).

Stacking is an alternative to ensembling in which we do not simply average the classifiers, but learn how to combine them – for IBCC, the combiner is a graphical model.



***Correlation and
large-scale simultaneous significance testing***
B. Efron

Multiple testing procedures assume the tests are independent: if not, they can be too conservative or liberal.

- Convert the test statistic to z scores;
- Fit a Gaussian to the middle of the distribution;
- Use this empirical null distribution instead of $N(0, 1)$ to compute the p -values.

In R, check the `locfdr` package.

***Homotopy parametric simplex method
for sparse learning***
H. Pang et al. (2017)

Many sparse learning problems,

$$\underset{\theta}{\operatorname{Argmin}} L(\theta) + \lambda \|\theta\|_1$$

or $\underset{\theta}{\operatorname{Argmin}} \|\theta\|_1 \text{ st } \|\nabla L(\theta)\|_\infty \leq \lambda$

(Dantzig selector, sparse regression, etc.) can be formulated as linear optimization problems. The *parametric simplex method* (PSM, or homotopy optimization) generalizes the simplex algorithm to solve those problem, while keeping the regularization parameter λ unspecified – the result is not a single solution, but a whole regularization path.

Voronoi treemaps
M. Balzer and O. Deussen (2005)

In a *centroidal Voronoi tessellation* (CVT), the generators are the centers of mass of their Voronoi cells. The “distances”

$$d(p_i, q) = \|p_i - q\| - w_i$$

$$d(p_i, q) = \|p_i - q\|^2 - w_i$$

define weighted Voronoi tessellations; the latter (power-weighted Voronoi diagram) has straight boundaries. To build a *Voronoi tree map*, start with a CVT (replace each generator with the center of mass of its cell and iterate a few times), adjust the weights in the direction suggested by the desired and current areas, and iterate.

Computing Voronoi treemaps
A. Nocaj and U. Brandes (2012)

The power (weighted Voronoi) diagram can be computed using a 3-dimensional convex hull algorithm.

***Certification of approximate roots of exact
polynomial systems***
A. Szanto (2016)

Bertini (non-free, but Bertini2, in development, will be GPL) solves large systems of algebraic equations (numerical algebraic geometry). An approximate zero z of

f is a point such that Newton’s iteration started at z converges to a root $\xi \in \mathbf{C}^n$ of f , with an upper bound R on $\|z - \xi\|$ and the guarantee that there is no other root in $B(z, R)$.

As an application, one can prove that, in \mathbf{R}^3 , seven (infinite) cylinders can touch each other.

Level set method
T.H. Colding and W.P Minicozzi (2016)

If the level sets of $v : \mathbf{R}^2 \rightarrow \mathbf{R}$ flow by mean curvature, then

$$v_t = |\nabla v| \operatorname{div} \left(\frac{\nabla v}{\|\nabla v\|} \right).$$

Hodge theory of matroids
K. Adiprasito et al. (2016)

The coefficients of the chromatic polynomial of a graph $\chi_G(q)$ (number of proper colourings of G using q colours form a log-concave sequence

$$a_i^2 \geq a_{i-1} a_{i+1}.$$

More generally, the number f_i of independent subsets of size i in a matroid is log-concave – this can be proved by defining a Hodge theory for matroids.

***TimescaleDB:
SQL made scalable for time series***

Postgres extension to store time series as tables (time, id₁, ..., id_n, v₁, ..., v_m), partitioned on (time, id).

***Implementing operational calculus
on programming spaces
for differentiable computing***
Ž. Sajovic (2017)

Automatic differentiation (AD) in C++, using templates.

High-dimensional discriminant analysis
C. Bouveyron et al.

Quadratic discriminant analysis (QDA) requires estimates of the variance matrix for each class: in high dimension, use regularized estimators, or make further assumptions, e.g., that the variance matrix only has two distinct eigenvalues.

Persistent topology of syntax
A. Port et al. (2015)

Application of TDA to language classification.

***Spin glass models of syntax
and language evolution***
K. Siva et al. (2015)

Modeling language interactions (from the number of bilingual editors for each pair of languages) using a *spin glass model* (aka *Potts model*, i.e., the *Ising model* with more than two states).

***Prevalance and recoverability of syntactic
parameters in sparse distributed memories***
J.J. Park et al. (2015)

Features to classify human languages (e.g., SVO order) can be found in:

- SSWL: syntactic structures of the world’s language;
- WAL: world atlas of language structures.

***Manopt, a matlab toolbox
for optimization on manifolds***
N. Boumal et al. (2014)

Describe the manifold (from a few predefined ones and a few operations: projection, quotient, product); provide the (Euclidean) derivatives; choose the method (trust region or conjugate gradient).

***Langevin and Hamiltonian based sequential
MCMC for efficient Bayesian filtering
in high-dimensional spaces***
F. Septier and G.W. Peters (2015)

Sequential Monte Carlo (SMC, particle filters) does not work well in high dimensions: instead of using importance weights, apply a few MCMC moves after each step.

***Information-geometric
Markov chain Monte Carlo methods
using diffusions***
S. Livingstone and M. Girolami (2014)

The Metropolis adjusted Langevin algorithm (MALA) can be generalized to Riemannian manifolds.

***Fast Langevin based algorithm
for MCMC in high dimension***
A. Durmus et al. (2016)

Other discretizations of the Langevin diffusion (SDE) give variants of MALA (Metropolis-adjusted Langevin algorithm) with faster convergence in higher dimensions.

Langevin-Euler moves

If $\pi(x) \propto \exp -U(x)$ then the solution of the Langevin diffusion SDE

$$dx = -\frac{1}{2} \frac{\partial U}{\partial x} dt + dW$$

follows the target distribution π .

PyCaMa: Python for cash management
F. Salas-Molina et al. (2017)

Given a set of bank accounts, desired transactions (from customers to suppliers), and transaction costs ($A \rightarrow B \rightarrow C$ may be cheaper than $A \rightarrow C$), finding the cheapest path is a linear problem.

***Downside risks
and the cross-section of asset returns***
A. Farago and R. Tédongap (2017)

The cross-section of returns can be explained with the covariance between stock returns and: market returns, change in market volatility, down times (boolean variable indicating either a fall in price or a rise in volatility), market returns $\times \mathbf{1}_{\text{down}}$, change in market volatility $\times \mathbf{1}_{\text{down}}$. Contrary to the Fama-French-Cahart model (which works better for stocks), this applies to all asset classes, including options and currencies.

Systematic tail risk
R.D.F. Harris et al. (2017)

The beta $\beta = \text{Cov}(R, R_m) / \text{Var } R_m$ and the correlation $\text{Cor}(R, R_m)$ can be turned into downside risk measures by conditioning on $R_m \leq \mu_m$, or replacing market returns R_m with their below-average part $\text{Min}\{R_m - \mu_m, 0\}$, or by replacing both asset and market returns with their below-average parts.

The extreme downside betas (EDB) and correlation (EDC) are obtained by replacing the means μ, μ_m with the 5% quantile. The extreme downside hedge (EDH) is the sensitivity of the forthcoming change in market value at risk, $\beta(R_t \sim \Delta \text{VaR}_{t \rightarrow t+1}^{\text{market}, 5\%})$, estimated from an AR(1)-GJR-GARCH(1,1) model with skewed Student innovations, on a 5-year moving window. There is a tail risk premium, visible if you compare the returns in $[t, t+1]$ and the risk estimated at time $t+1$, but it is not investible.

***Conditional asset pricing
in international equity markets***
T. Huynh (2017)

The returns of an equal-weighted momentum-value portfolio can be explained using the “lagged component” betas as instrumental variables.

The time-varying betas of a portfolio (on a moving window) can be estimated directly, by considering the portfolio as a single asset (“lagged portfolio” betas) or as a portfolio-weighted average of stock-level betas (“lagged component” betas).

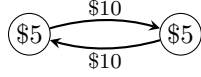
***Contagion in financial systems:
a Bayesian network approach***
C. Chong and C. Klüppelberg (2017)

Tightly interconnected firms are robust to small shocks, but the network amplifies large ones (phase

transition, non-monotonic effect). Model firms as:

$$\begin{aligned} X_i &\sim N(\mu_i, \sigma^2) && \text{assets} \\ L_{ij} &&& \text{liabilities} \\ E_i &= X_i + \sum_{j \text{ survives}} L_{ij} - \sum_j L_{ji} && \text{equity.} \end{aligned}$$

If the redemption graph has no cycles, the defaulting and surviving firms are well-defined but, if not, there are multiple solutions with non-zero probability. For instance, with



both survive if netting is possible, but both default if not.

Two solutions are obtained by progressively building the list of defaulting (resp. surviving) firms. Using those rules, one can extend the redemption graph into a directed acyclic graph (DAG), by creating N copies of each vertex, corresponding to the iterations of the algorithm. Bayesian network tools are then available: independence detection, systemic impact measures, default probabilities, etc.

Instrumented principal component analysis **B. Kelly et al. (2017)**

Factor models combine stock-specific but constant loadings β with common, dynamic factors f :

$$y_{nt} = \sum_k \beta_{nk} f_{kt} + \varepsilon_{nt}.$$

Dynamic factor models usually assume a simple model for the changes of β , e.g., a random walk:

$$y_{nt} = \sum_k \beta_{nk} f_{kt} + \varepsilon_{nt}$$

$$\beta_{n,k,t} = \beta_{n,k,t-1} + \eta_{n,k,t}$$

Instead, one can use additional data (time-changing, stock-specific “instrumental” variables Z) to predict β :

$$y_{nt} = \sum_k \beta_{nk} f_{kt} + \varepsilon_{nt}$$

$$\beta_{n,k,t} = \sum_\ell Z_{k\ell,t} \Gamma_{\ell,k} + \eta_{n,k,t}.$$

The model can be estimated as follows:

- Orthonormalize the instruments so that, for each t , $N^{-1} Z_t' Z_t = I_L$;
- Set $x_{\ell,t} = N^{-1} \sum_n Z_{n,\ell,t-1} y_{n,t}$;
- Let $\Gamma_{\cdot,1}, \dots, \Gamma_{\cdot,k}$ be the k eigenvectors of $N^{-1} X' X$ for the K largest eigenvalues, normalized ($\Gamma' \Gamma = I_K$) and signed (so that the first non-zero entry of each column be positive);
- Set $f_{k,t} = \sum_\ell \Gamma_{\ell,k} x_{\ell,t}$.

If $y_{n,t}$ are stock returns and $z_{n,\ell,t}$ stock characteristics, then $x_{\ell,t}$ are the returns of portfolios defined using the (raw) instrumental variables Z as weights – factor-mimicking portfolios usually discretize them first.

Second-order optimization over the multivariate Gaussian distribution **L. Malagò and G. Pistone (2016)**

On a Riemannian manifold, the gradient and the Hessian, used in first- and second-order optimization algorithms (e.g., gradient descent, Newton) become the *natural gradient* and the *Riemannian Hessian*. For statistical manifolds and their three connections (Levi-Civita, exponential, mixture), there are actually three Hessians.

Stochastic relaxation replaces

$$x^* = \underset{x \in \mathbf{R}^n}{\text{Argmin}} f(x)$$

with

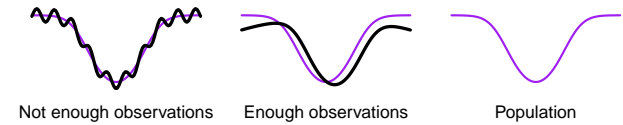
$$(x^*, 0) = \underset{\mu, V}{\text{Argmin}} \mathbb{E}_{X \sim N(\mu, V)} [f(X)].$$

The landscape of empirical risk for non-convex losses **S. Mei et al. (2016)**

Given enough observations ($\Omega(d \log d)$ in dimension d), non-convex loss functions, e.g.,

$$\frac{1}{n} \sum (y_i - \sigma \langle w, x_i \rangle)^2$$

for binary classification, have a unique minimizer.



Information geometry of neural networks **S.I. Amari**

Natural gradient descent, for neural nets, with an approximation of the Riemannian metric, avoids plateaus.

Hyperparameter optimization with approximate gradient **F. Pedregosa (2016)**

Hyperparameter optimization is a bilevel optimization problem

$$\begin{aligned} \text{Find} & \quad \lambda, \beta^* \\ \text{To minimize} & \quad f(\lambda) = g(\beta^*, \lambda) \\ \text{Such that} & \quad \beta^* = \underset{\beta}{\text{Argmin}} h(\beta, \lambda) \end{aligned}$$

where

λ : hyperparameters
 β : parameters
 h : loss
 g : cross-validation loss.

It can be reformulated as a constrained optimization

$$\begin{array}{ll} \text{Find} & \lambda, \beta \\ \text{To minimize} & f(\lambda) = g(\beta, \lambda) \\ \text{Such that} & \nabla_{\beta} h(\beta, \lambda) = 0. \end{array}$$

The gradient can be computed:

$$\begin{aligned} \frac{df}{d\lambda} &= \frac{\partial \beta}{\partial \lambda} \frac{\partial g}{\partial \beta} + \frac{\partial g}{\partial \lambda} \\ \frac{\partial}{\partial \beta} [\nabla_{\beta} h(\beta, \lambda)] &= 0 \\ \nabla f &= \nabla_{\lambda} g - (\nabla_{\beta, \lambda} g)' (\nabla_{\beta, \beta} g)^{-1} \nabla_{\beta} g. \end{aligned}$$

Approximations for β^* , ∇g and $\nabla^2 g$ are good enough.

**Mapping estimation
for discrete optimal transport**
M. Perrot et al. (2016)

Optimal transport, in particular the *Wasserstein distance*, is used in machine learning, as a divergence to compare distributions, to interpolate between distributions, to compute means, barycenters or PCA in spaces of probabilities.

Given probability spaces (X, μ) and (Y, ν) and a cost function $c : X \times Y \rightarrow \mathbf{R}$, the **Monge problem** asks for a function $T : X \rightarrow Y$ such that $T_{\#}\mu = \nu$ ($T_{\#}$ is the push-forward) minimizing $\int_X c(x, T(x)) d\mu(x)$. The **Kantorovitch** relaxation asks instead for a probability distribution γ on $X \times Y$ with marginals μ and ν minimizing $\int_{X \times Y} c(x, y) d\gamma(x, y)$. For discrete distributions, in particular, for empirical distributions, this is a linear program.

Jointly learn the *probabilistic coupling* γ (from the empirical distributions) and the *barycentric mapping* T (in a limited set of functions, e.g., linear, or using some kernel) with a penalty (e.g., to ensure T is close to the identity).

While transfer learning transforms a model $X \rightarrow Y$ into a model $X \rightarrow Y'$ with a new target (codomain), **domain adaptation** transforms it to a model $X' \rightarrow Y$ with a different source space (domain).

In image processing, before inpainting (Poisson image editing), one can force the gradient distribution of the source image to match that of the target image (using a random sample of 500 gradients on each side).

**Learning to learn, from transfer learning
to domain adaptation: a unifying perspective**
N. Patricia and B. Caputo

Supervised word mover's distance
G. Huang et al. (2016)

Combine the word mover's distance (the distance between two documents is the cost of the optimal transport between their (vector-embedded) bags of words) with metric learning (use the distance minimizing the

leave-out-one (LOO) k -nearest neighbour error when predicting document labels).

**SAGA: a fast incremental gradient method
with support for non-strongly convex
composite objectives**
A. Defazio et al. (2014)

To minimize a smooth convex function $f(x) = \frac{1}{n} \sum_i f_i(x)$, *variance-reduced stochastic gradient descent* uses updates of the form

$$x \leftarrow x - \gamma \frac{1}{n} \left[f'_i(x) - f'_i(x^*) + \sum_j f'_j(x^*) \right] \quad (\text{SAG})$$

$$\text{or } x \leftarrow x - \gamma \left[f'_i(x) - f'_i(x^*) + \frac{1}{n} \sum_j f'_j(x^*) \right] \quad (\text{SVRG})$$

where x^* and $\sum f'_i(x^*)$ are updated once in a while.

SAGA keeps track of n points $(x_i)_{1 \leq i \leq n}$ and the derivatives of the corresponding terms:

$$x \leftarrow x - \gamma \left[f'_i(x) - f'_i(x^*) + \frac{1}{n} \sum_j f'_j(x_j) \right];$$

one of them is updated at each step with the current value of x .

**Accelerating stochastic gradient descent
using predictive variance reduction**
R. Johnson and T. Zhang (2013)

To reduce variance in stochastic gradient descent (SVRG), approximate the gradient of $f = \sum f_i$ as

$$\nabla f(x) \approx \nabla f(x^*) - \nabla f_i(x^*) + \nabla f_i(x).$$

Faster asynchronous SGD
A. Odena

Distributed stochastic gradient descent (SGD) ignores synchronization problems, but keeps track of gradient staleness, either with the number of updates skipped, or with the difference between the gradient computed and the gradient that would have been computed (on the same minibatch) with the latest parameters, and uses it to fine-tune the learning rate (not unlike RMS-prop).

Fast and provably good seedings for k-means
O. Bachem et al. (2016)

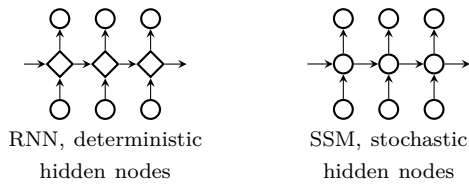
Kmeans++ seeding does not scale to large datasets. Instead of sampling each point exactly from $p(x) \propto d(x, C)^2$, use MCMC, with the exact distribution after the first point (mixed with the uniform distribution) as proposal distribution, instead of a uniform distribution.

**Phased LSTM: accelerating recurrent network
training for long or event-based sequences**
A. Neil et al. (2016)

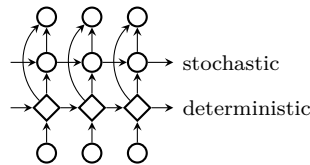
By allowing neurons to update at different frequencies (by adding a “time gate”), LSTMs can deal with irregularly-spaced signals, e.g., event streams.

Sequential neural models with stochastic layers
M. Fraccaro et al. (2016)

Recurrent neural nets (RNN) and state space models (SSM) are eerily similar.



Stochastic RNNs stack them – not unlike a variational auto-encoder (VAE) at each step.



Improved techniques for training GANs
T. Salimans et al. (2016)

Training a GAN is a Nash equilibrium problem

$$\theta_D \in \text{Argmin } J_D(\theta_D, \theta_G)$$

$$\theta_G \in \text{Argmin } J_G(\theta_D, \theta_G) :$$

gradient descent can fail to converge.

- To prevent the generator from overfitting the discriminator, have the generator generate data whose statistics match those of the real data, e.g., the expected values of the features on an intermediate layer of the discriminator (feature matching);
- Include a term for historical averaging

$$\left\| \theta - \frac{1}{t} \sum_{1 \leq i \leq t} \theta_i \right\|^2 ;$$

- To prevent the generator from always emitting the same point, allow the discriminator to look at several examples (minibatch discrimination);
- Replace the 0 and 1 targets with smoothed values (label smoothing);
- Batch normalize using the statistics of a fixed reference batch (virtual batch renormalization).

DeepMath – deep sequence models for premise selection
A.A. Alemi et al. (2016)

Using the Mizar mathematical library (the library is free, the theorem prover is not) of formal proofs (of real-world theorems: Hahn-Banach, Brouwer, Borsuk-Ulam, Gödel, etc.), train a RNN and/or a CNN, with LSTM/GRU, from character or word embeddings, to select which axioms will be useful in proving a given conjecture, to guide automated theorem provers.

Learning to pivot with adversarial networks
G. Louppe et al.

To make a model f of $Y \sim X$ robust to a latent parameter Z (e.g., model specification),

$$\begin{array}{ll} \text{Find} & f \\ \text{Such that} & Y \approx f(X) \\ \text{and} & f(X) \perp\!\!\!\perp Z \end{array}$$

using an adversarial network:

- Learn f so that $y \approx f(x)$ and g performs poorly;
- Learn g to predict z from $f(x)$.

Deep networks with stochastic depth
G. Huang et al. (2016)

During training, randomly drop a subset of layers.

Exponential family embeddings
M. Rudolph et al. (2016)

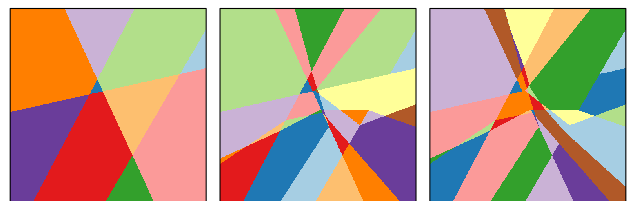
Dimension reduction with word vector embedding looks for a map $f : \text{Words} \rightarrow \mathbf{R}^n$ minimizing $\text{Loss}(f(\text{word}), f(\text{surrounding words}))$. One can replace the word and the surrounding words, and their discrete distributions, with exponential distributions conditioned on a “context”, for instance:

- A Gaussian embedding can be used to model neuron activity, conditioned on the activity of nearby neurons;
- A *Poisson embedding* can model items conditioned on other items from the same shopping basket.

On the expressive power of deep neural networks
R. Raghu et al.

To measure the expressivity of a neural net, check how the length of a 1-dimensional trajectory in the input space changes as it propagates through the network; for instance, for the output node, look at the number of sign changes.

The regimes in which the neurons are (ReLU have two regimes, zero and linear; hard-tanh have three, linear and saturated at ± 1) define the “activation patterns” of a network (binary strings, for ReLUs). For ReLU networks, the activation patterns subdivide the input space into convex polytopes.



Matrix factorization using window sampling and negative sampling for improved word representations

A. Salle et al.

Let M_{wc} be the cooccurrence matrix between words w and context words c ,

$$\text{PMI}_{wc} = \log \frac{M_{mw} M_{\cdot, \cdot}}{M_{w, \cdot} M_{\cdot, c}}$$

the pointwise mutual information matrix and PPMI the positive semidefinite matrix obtained by zeroing out negative eigenvalues. These matrices have been used to define word embeddings as follows:

- PPMI-SVD factorizes $\text{PPMI}_{wc} \approx W_w W'_c$ using a truncated SVD, $\text{PPMI} = U \Sigma V'$, $W = U \Sigma^{1/2}$;
- GloVe factorizes $\log M$ by minimizing

$$\sum_{w,c} f(M_{wc})(W_w W'_c + b_w + \tilde{b}_c - \log M_{wc})^2;$$

where the weights are

$$f(x) = \begin{cases} (x/x_{\max})^\beta & \text{if } x \leq x_{\max} \\ 1 & \text{otherwise;} \end{cases}$$

- Skipgram with negative sampling factorizes PMI by minimizing

$$\sum_{w,c} (W_w W'_c - \text{PMI}_{wc})^2 - E_{w'} (W_w W'_{w'} - \text{PMI}_{ww'})^2;$$

- LexVec is similar but factors PPMI.

On multiplicative integration with recurrent neural networks

Y. Wu et al.

Replace the computational blocks $\phi(Wx + Uz + b)$ with $\phi(Wx \odot Uz + b)$. [This is one of the ideas behind LSTMs, but it can be used separately.]

Tagger: deep unsupervised perceptual grouping

K. Greff et al.

To segment an image, look for groupings of pixels for which denoising autoencoders easily reconstruct the image withing each group.

Neural networks with differentiable structure

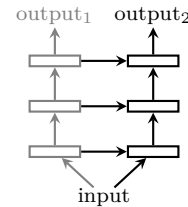
T. Miconi

To select network structure via gradient descent, make it differentiable by multiplying the output of each neuron with a scalar factor (e.g., one for each layer), and add an L^1 penalty for those new parameters to encourage sparsity.

Progressive neural networks

A.A. Rusu et al.

To learn a task related to an already-learned one, consider a new neural net, parallel to the first one, randomly initialized, with lateral connections. To keep the model scalable (task $n + 1$ has connections from tasks 1 through n), use dimension reduction (single-layer network) for those lateral connections.



Visualizing deep convolutional neural networks using natural pre-images

A. Mahendran and A. Vedaldi (2016)

To understand what a neural network (CMM for computer vision or image recognition) sees:

- Compute the inverse of a representation, i.e., look at images sharing the same representation;
- Maximize a certain component (cf. the “cat neuron”);
- Exaggerate the patterns present in an image (deep dream).

Inception-v4, Inception-ResNet and the impact of residual connections on learning

C. Szegedy et al.

Adding residual connections to Inception networks speeds up training.

Question answering via integer programming over semi-structured knowledge

D. Khashabi et al.

Factoid question answering systems just look for an already written answer in a large corpus, but cannot combine separate pieces of information to create a new, never-seen sentence. Given tables of knowledge (65, for a total of 5000 rows, origin unknown), integer programming can find a way to join them to best answer a given question.

Do deep convolutional nets really need to be deep (or even convolutional)?

G. Urban et al.

The results of a deep convolutional network can be reproduced by a shallower network, but it still needs to be deep (one hidden layer is not enough) and convolutional.

Tweet2vec: character-based distributed representations for social media
B. Dhingra et al.

Character-level bidirectional LSTMs or GRUs can learn vector embeddings that generalize well to rare, abbreviated or misspelt words, frequent on Twitter.

Detecting relative anomaly
R. Neuberg and Y. Shi

To detect anomalies:

- From the distance matrix, compute a similarity matrix via a Gaussian kernel and use its first eigenvector as a measure of centrality (similar to PageRank); with the normalized similarity matrix, this is actually equivalent to a kernel density estimator.
- Consider an observation highly normal if its degree is in the top $\alpha\%$; label an observation anomalous if the length of the shortest path to the highly normal observations is high (in the top $\beta\%$)

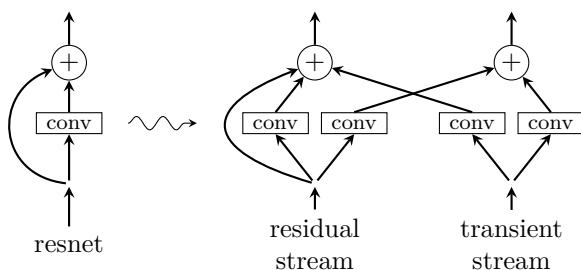
Unifying count-based exploration and intrinsic motivation
M.G. Bellemare et al.

Practical applications of reinforcement learning still rely on ϵ -greedy policies for exploration. *Intrinsic motivation* (IM) achieves exploration using *novelty* signals, e.g., prediction error or value error. Sequential density models (probability distributions on the set of sequences of states) can be used to define pseudo-counts: count-based reinforcement learning algorithms then apply (experience replay, actor-critic, etc.).

Model-free episodic control
C. Blundell et al.

Gradient-based reinforcement learning is slow: to speed it up, remember (not learn) successful strategies and re-enact them when the agent encounters a similar state. (In humans, the hippocampus does that.)

Resnet in Resnet: generalizing residual architectures
S. Targ et al.



Combination of two-dimensional cochleogram and spectrogram features for deep learning-based ASR
A. Tjandra et al. (2015)

The **cochleogram**, a convolution with the *gammatone filter*

$$g(t) = t^{n-1} e^{-2\pi b t} \cos(2\pi \omega t)$$

n : filter order
 $b = 110 \omega + 25.2$: bandwidth

is an alternative to MFCC (Mel frequency cepstral coefficients) features.

Visually identifying rank
D.F. Fouhey et al.

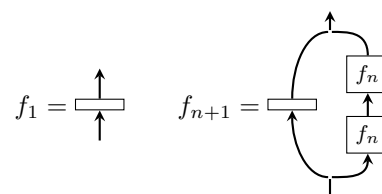
Can CNNs (or SVMs with VLFeat features) recognize the rank of a matrix? Results are not as disastrous as you may think.

High-dimensional probability estimation with deep density models
O. Rippel and R.P. Adams

Dimension reduction of a dataset Y in \mathcal{Y} to a simpler space \mathcal{X} can be obtained by two functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ and $g : \mathcal{Y} \rightarrow \mathcal{X}$ (neural nets with sigmoid functions) minimizing the divergence between the marginals of $g(Y)$ and the desired marginals (Beta distribution with $\alpha \gg 1$) with a penalty for poorly conditioned f (we want it invertible) and another one to ensure $g \approx f^{-1}$.

FractalNet: ultra-deep neural networks without residuals
G. Larson et al.

Fractal networks are defined recursively,



where f is, for instance, a convolutional layer, and trained with *drop path* (each join drops each input with some probability, but keeps at least one) and global (keep only one path, with layers in the same column).

Random rotation ensembles
R. Blaser and P. Fryzlewicz (2016)

To reduce the artifacts (and the number of trees needed) in ensembles of decision trees, randomly rotate the feature space before training the base learners.

***Accelerating Eulerian fluid simulation
with convolutional networks***
J. Tompson et al.

Machine learning and convolutional nets can help solve PDEs.

***Compressing deep convolutional networks
using vector quantization***
Y. Gong et al. (2014)

Compress dense weight matrices with:

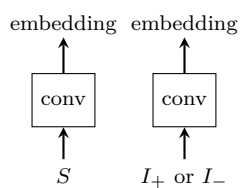
- Matrix factorization;
- Scalar quantization, by replacing the matrix entries with their signs, or by clustering them with k -means;
- Product quantization: split the matrix into blocks of columns; for each block, cluster the rows with k -means;
- Residual quantization: cluster the rows, then the residuals, and iterate a few times.

***The sketchy database:
learning to retrieve badly drawn bunnies***
P. Sangkloy et al.

A **siamese network** is a pair of convnets detecting if two inputs are similar or not; they use a contrastive loss.

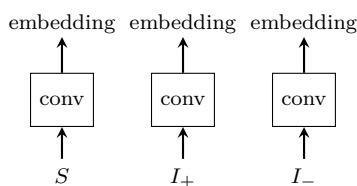
$$\text{Loss} = \lambda d(S, I_+) + (1 - \lambda)(m - d(S, I_-))_+$$

$\lambda = 0$ or 1
 S : source image
 I_+ : matching image
 I_- : non-matching image
 m : margin



Triplet networks are similar, but their loss is of the form “ S is closer to I_+ than to I_- ”.

$$\text{Loss} = (d(S, I_+) - d(S, I_-))_+$$



For a dataset of sketch-photo pairs, the two (or two of the three) convnets are different.

***Interaction networks for learning
about objects, relations and physics***

To design a data-driven physics engine, learn:

- Object-centric functions, which map the state of an object at time t to its state at time $t + 1$ (or, rather, a state increment);
- Relation-centric functions, which map the states of two objects and the characteristics of the relation (e.g., the spring constant) to their state increments;
- An aggregation function, which combines the state increments.

***On the modeling of musical solos
as complex networks***
S. Ferretti (2016)

From a piece of music, create a network, with pitch-duration pairs as nodes (you can also add rests and chords), and directed arcs to denote succession; the usual graph statistics (degree distribution, centralities, etc.) can help classify the piece.

***A first look at music composing using LSTM
recurrent neural networks***
D. Eck and J. Schmidhuber

LSTMs can generate the global structure RNNs cannot capture.

***DeepBach: a steerable model
for Bach chorales generation***
G. Hadjeres and F. Pachet

Represent the data as 6 sequences, corresponding to the 4 voices (either the midi pitch or a “hold” symbol), the beat (12341234...) and the presence of a fermata; the last two are fixed.

Model the data as $v_i = f(v_{<i}, v_{>i})$, where f is a neural net, processing $v_{<i}$ and $v_{>i}$ with LSTMs (in opposite directions).

Generate new chorales with Gibbs sampling.

The **pseudo-log-likelihood** is a Gibbs-inspired approximation of the log-likelihood.

$$\begin{aligned} \text{Log-likelihood} &: \log P(V_1, \dots, V_n | \theta) \\ \text{Pseudo-log-likelihood} &: \sum_i \log P(V_i | V_{\neq i}, \theta) \end{aligned}$$

***An information-theoretic approach
to machine-oriented music summarization***
F. Raposo et al.

Text summarization extracts relevant and diverse (bits of) sequences; for music, this can create harsh discontinuities. Compute MFCC features on a moving window, model them as a (mixture of) multivariate Gaussians, use this distribution as a (machine-oriented) summary, and somehow turn it into a human oriented summary. Implementation in C++ using openSMILE (feature extraction) and Marsyas (synthesis).

***Text extraction from the web
via text-to-tag ratio***
T. Weninger and W.H. Hsu

To detect relevant text in the midst of today's HTML pages tag soup:

- Remove empty lines, comments, `<script>` and `<style>` blocks;
- In each line, count the number of (non-tag) characters, the number of tags, the ratio $\text{text}/\max(1, \text{tag})$;
- Apply k -means on smoothed ratios and pick the cluster with the largest average ratio,
- Or pick lines whose smoothed ratio is above a one standard deviation threshold,
- Or apply some technical analysis indicator.

This could be combined with

- Other line metrics, e.g., entropy;
- Template detection (often, pages on the same web site belong to the same or a few templates);
- Machine learning algorithms to detect boilerplate (not unlike span detection).

CETR: content extraction via tag ratios
T. Weninger et al. (2010)

Instead of clustering the 1-dimensional text-to-tag ratio x , cluster $(x, |\dot{x}|)$ (after smoothing).

***A machine learning approach
to webpage content extraction***
J. Yao and X. Zuo

After discarding `<nav>` and `<aside>` elements (contents should be in `<article>`), consider blocks rather than lines and train a machine learning classifier (SVM, naive Bayes) using the following features: number of words, average sentence length, text density, link density (and the ratios wrt the previous block), block number (rescaled to $[0, 1]$), id and class attributes.

***CrossCat: a fully Bayesian nonparametric
method for analyzing heterogeneous,
high-dimensional data***
V. Mansinghka et al.

CrossCat is a hierarchical nonparametric Bayesian model for tabular data, used by BayesDB, whose generative process looks like:

- Partition the columns, using a Chinese restaurant process (CRP);
- Likewise partition the rows;
- Sample the parameters for each column (variable) in each intersection;
- Sample the rows in each intersection.

***BayesDB: a probabilistic programming system
for querying the probable implications of data***
V. Mansinghka et al.

When you insert data in a table, BayesDB estimates a model that allows it to:

- Sample new rows from the resulting joint or conditional distribution;
- Compute the likelihood of each row, to detect outliers;
- Retrieve similar rows;
- Compute the mutual information (MI) between pairs of variables (or, rather, the probability that the posterior MI is nonzero), which is better than correlation at detecting dependencies;
- Infer missing values.

The univariate models (“data types”) are:

- Numeric: normal-gamma;
- Count: Poisson-gamma;
- Binary: asymmetric Beta-Bernoulli;
- Categorical: multinomial with symmetric Dirichlet prior.

***Identifying almost identical files using
context-triggered piecewise hashing***
J. Kornblum (2006)

To check if two files are almost identical:

- Compute a hash on a rolling window of moderate size and note the positions at which it takes a specific value (trigger); discard the hashes;
- Compute a hash for the parts between two triggers.

[One could also compute a longer rolling hash and only keep the values if the first bytes have a specified value.] This is implemented in `ssdeep`; similar algorithms include `tlsh` and `sdhash`. Some use CTPH as features in machine learning algorithms to detect or cluster spam, malware, bot control centers, etc.

An introduction to topology
M.J. Dominus (2010)

A *valuation* for classical logic is a map

$$v : \{\text{formulas}\} \longrightarrow \{\top, \perp\}.$$

Instead of $\{\top, \perp\}$, one can use $(\mathcal{P}(X), \cap, \cup)$ or any other boolean algebra. *Tautologies* are formulas whose value is \top (or $X \in \mathcal{P}(X)$).

For *intuitionistic logic*, where $\neg p \vee p$ is not a theorem, one replaces Boolean algebras with sufficiently large *Heyting algebras*, e.g., $\mathcal{P}(\mathbf{R})$:

$$\begin{aligned} A \vee B &= A \cup B \\ A \wedge B &= A \cap B \\ A \rightarrow B &= (A^c \cup B)^\circ \\ \neg A &= A^{c^\circ}. \end{aligned}$$

The *compactness principle* (i.e., Tychonov's theorem) says that if any finite subset of a set of axioms has a model, then so does the whole set – for instance, with the real numbers axioms and $\varepsilon < 1$, $\varepsilon < 1/2$, $\varepsilon < 1/3$, ..., $\varepsilon > 0$, this allows for infinitesimals.

**Tensor rank and the ill-posedness
of the best low-rank approximation problem**
V. de Silva and L.H. Lim (2006)

Many properties of the rank of a matrix do not generalize to tensors of order $k \geq 3$:

- The *outer product rank*

$$\text{rank}_{\otimes} A = \text{Min} \left\{ n : \exists x_{ij} \ A = \sum_{i=1}^n x_{1i} \otimes \cdots \otimes x_{ki} \right\}$$

and the multilinear rank rank_{\boxplus} (k -tuple of the dimensions of the spaces spanned by the columns of the k matricizations of the tensor) do not coincide.

- A tensor in $\mathbf{R}^{d_1 \times \cdots \times d_k}$ can have rank greater than $\text{Min}(d_1, \dots, d_k)$.
- Tensor rank is not upper semi-continuous; many tensors (a non-negligible set) do not have a best rank r approximation: for instance, the rank-3 tensor

$$A = x_1 \otimes x_2 \otimes y_3 + x_1 \otimes y_2 \otimes x_3 + y_1 \otimes x_2 \otimes x_3$$

can be approximated arbitrarily close by the rank-2 tensors

$$A_n = n \left(x_1 + \frac{1}{n} y_1 \right) \otimes \left(x_2 + \frac{1}{n} y_2 \right) \otimes \left(x_2 + \frac{1}{n} y_2 \right) \\ - n x_1 \otimes x_2 \otimes x_3.$$

**Scalable latent tree model
and its application to health analytics**
F. Huang et al.

The *multivariate information distance*

$$d(X, Y) = -\log \frac{\prod \sigma_i [\text{Cov}(X, Y)]}{\sqrt{\det \text{Cov}(X, X) \det \text{Cov}(Y, Y)}}$$

where the σ_i are the top- k singular values generalizes the correlation to multivariate random variables (on a tree, this distance is additive).

To learn a *latent tree graphical model* (an undirected graphical model, with no loops, and both observed and unobserved variables), compute the matrix of multivariate information distances; build the corresponding minimum spanning tree; find triplets of nodes for which the distance is additive: this indicates the presence of a hidden node; add it and iterate. To estimate the model parameters, compute a low rank tensor decomposition of the third moment of triplets of variables $E[X_1 \otimes X_2 \otimes X_3]$.

**High-dimensional neural spike train analysis
with generalized count
linear dynamical systems**
Y. Gai et al.

The Poisson distribution $p(k) \propto \lambda^{-k} e^{-k} / k!$ is a special case of the **generalized count** (GC) distribution, $p(k) = e^{\theta k + g(k)} / k!$ which can account for over- or under-dispersion.

It can be used in latent variable models $y_{ij}|x_j \sim \text{Poisson}(\exp c'_{ij} x_j + d)$ where the state x_j is modeled by a Gaussian dynamic system $x_{j,t+1}|x_{j,t} \sim N(Ax_{j,t} + b, Q)$.

Large scale unusual time series detection
R.J. Hyndman et al. (2015)

To detect outliers in a dataset of time series: compute features for each time series (mean, variance, autocorrelation, trend, linearity, curvature, seasonality, peakedness, troughiness, spectral entropy, lumpiness, spikiness, level shift, variance change, flat spots, crossing points, Kullback-Leibler, etc. – scagnostics), reduce the dimension (PCA), and use 2-dimensional outlier detection algorithms (alpha-hull or density-based).

**Topology data analysis of critical transitions
in financial networks**
M. Gidea (2017)

Compute the correlation matrix of asset returns on a moving window, convert it to a distance $d_{ij} = \sqrt{1 - c_{ij}}$, compute the persistence diagram of the corresponding Rips filtration (with `TDA::ripsDiag`) and look at the Wasserstein or bottleneck distance between two dates: they may help forecast crises.

[How different is it from the L^2 distance between the correlation matrices?]

**Network-based anomaly detection
for insider trading**
A. Kulkarni et al.

Build a network with insiders (of the same company) as nodes and an edge whenever the length of the *longest common subsequence* (LCS) of ordered trade dates exceeds some threshold (5 for sales, 10 for purchases). Look at the *egonets*, *i.e.*, the subgraphs induced by the nodes at distance at most one from a given node. Identify outliers by the residuals of $\log \#E \sim \log \#V$ for the *egonets*, and the local outlier factor (LOF) comparing the density of a node with that of its neighbours; they are hubs or bridges between highly connected components.

The same approach applies to hypergraphs (a hyper-edge between a set of insiders if the length of the LCS of trade dates exceeds some threshold); the profit of the transactions in those hyperedges tends to be positive.

Graph-based anomaly detection
C.C. Noble and D.J. Cook (2003)

To detect repetitive patterns (substructures) in a graph, compare the description length of the graph before and after contracting this substructure

$$\text{DL}(G) - (\text{DL}(G|S) + \text{DL}(S)).$$

Compress the most frequent substructure and iterate (“Subdue” algorithm).

To detect anomalies in a graph. compute

$$A = 1 - \frac{1}{n} \sum_{i=1}^n (n - i + 1) \frac{\text{DL}_{i-1}(S) - \text{DL}_i(S)}{\text{DL}_0(S)}$$

for each substructure S , where $DL_i(S)$ is the description length of S after i steps of the algorithm.

To detect anomalies in a graph, one could also find substructures S such that

$$\text{size of } S \times \text{number of instances of } S$$

be small.

RolX: Structural role extraction and mining in large graphs
K. Henderson et al. (2012)

To identify “roles” in a network (e.g., star center, clique member, peripheral node, etc.), compute node features (degree, number of within-egonet edges, average and maximum neighbour degree, number of triangles, etc.) and soft-cluster them with nonnegative matrix factorization (NMF).

It's who you know: graph mining using recursive structural features
K. Henderson et al. (2010)

Recursive graph features are built as follows:

- Number of within-egonet edges;
- Number of edges leaving or entering the egonet;
- Mean or sum of a feature over the egonet.

Prune the features by discarding those similar to previous ones; prune them further by focusing on those with good predictive power on node attributes.

Applications include graph de-anonymization, graph isomorphism, graph-based anomaly detection.

Graph-based anomaly detection and description: a survey
K. Akoglu et al.

Statistical industry classification
Z. Kakushadze and W. Yu (2017)

To cluster stocks, use excess returns, normalized by the squared volatility (rather than the volatility, to reduce the influence of volatile stocks – in practice, consider something like $r/\sigma \text{Max}(\bar{\sigma}, \sigma)$ to avoid creating a similar problem with low-volatility stocks).

To choose the number of clusters, use the *effective rank* of the correlation matrix:

$\lambda_1, \dots, \lambda_n$ eigenvalues

$$p_i = \frac{\lambda_i}{\sum \lambda_j}$$

$$H = -\sum p_i \log p_i \text{ (spectral) entropy}$$

$$\text{erank } C = \exp H.$$

The effective rank: a measure of effective dimensionality
O. Roy and M. Vetterli (2007)

Instead of minimizing the rank of a matrix, minimize its **effective rank**: compute its singular values $\sigma_1 \geq \dots \geq \sigma_m \geq 0$, rescale them $p_i = \sigma_i / \sum \sigma_j$, compute the *spectral entropy* $H = -\sum p_i \log p_i$ and take its exponential.

Statistical risk models
Z. Kakushadze and W. Yu (2017)

To estimate the number of factors in a statistical risk model, use the effective rank: compute the eigenvalues, rescale them to have a probability distribution, compute its entropy, and the corresponding effective number of values.

Since the “market” factor dominates, the result is often close to 1: if this is not desirable, remove the largest eigenvalue and add 1 back at the end.

Optimization can be approximated with a weighted regression on the first principal components, with the inverse of the specific variances as weights.

Decoding Chinese stock market returns: three-state hidden semi-Markov model
Z. Liu and S. Wang (2016)

The *Taylor effect* is yet another stylized fact: $\text{Cor}(|r_t|^\theta, |r_{t-\tau}|^\theta)$ is maximum for $\theta = 1$.

In a hidden Markov model (HMM), the sojourn time follows a geometric distribution. **Hidden semi-Markov models** (HSMM, estimated with the EM algorithm) relax that restriction by separately modeling the sojourn time and the state transitions.

Spatial dependence in asset pricing models
B. Zhu and S. Milcheva

To price real estate firms, add the physical distance between their properties to the CAPM or the factor model used.

$$r_i = \alpha_i + \rho \sum_{j \neq i} w_{ij} r_j + \sum_k \beta_{ik} f_k + \varepsilon_i, \quad w_{ij} \propto d_{ij}^{-1} \text{ or } d_{ij}^{-2}$$

Time series copulas for heteroskedastic data
R. Loaiza-Maya et al. (2016)

In presence of heteroskedasticity, the dependence structure of a time series (X_{t-1}, X_t) cannot be described by the common parametric copulas: the observations gather in the four corners, or along an “X”, instead of one or two corners and along a diagonal.

One can use *mixtures of rotated copulas* (or D-vines (drawable vines) of them, in higher dimensions).

DEA portfolio modeling
S.M. Gasser (2016)

Data envelopment analysis (DEA) is often used as a screening step, to select stocks according to multiple criteria, before portfolio optimization. Instead, one can directly use the DEA efficiency score to compute the portfolio weights (just rescale them).

Deep learning for finance: deep portfolios
J.B. Heaton et al. (2016)

Use autoencoders for non-linear PCA regression to replicate an index, or to explain returns (as with the CAPM or factor models).

The volatility of bitcoin
A. Urquhart (2017)

The *heterogeneous auto-regressive* (HAR) model is an alternative to GARCH models with high frequency data:

$$\sigma_{[t,t+1]} \sim \sigma_{[t-1,t]} + \sigma_{[t-7,t]} + \sigma_{[t-30,t]}.$$

Efficient algorithms for computing risk parity portfolio weights
D. Chaves et al. (2012)

The risk parity portfolio can be computed using Newton's method to solve

$$\begin{aligned} Vw &= \lambda w^{-1} \\ w'1 &= 1 \end{aligned}$$

(the unknowns are λ and w),

$$\begin{aligned} J &= \begin{pmatrix} V + \lambda \text{diag}(w^{-2}) & -w^{-1} \\ 1' & 0 \end{pmatrix} \\ \begin{pmatrix} w \\ \lambda \end{pmatrix} &\leftarrow \begin{pmatrix} w \\ \lambda \end{pmatrix} - J^{-1} \begin{pmatrix} Vw - \lambda w^{-1} \\ 1'w - 1 \end{pmatrix} \end{aligned}$$

or by iterating

$$\begin{aligned} w_i &\leftarrow \frac{\beta_i^{-1}}{\sum \beta_j^{-1}} \\ \beta_i &\leftarrow \frac{(Vw)_i}{w'Vw}. \end{aligned}$$

These algorithms are not guaranteed to converge.

A fast algorithm for computing high-dimensional risk parity portfolios
T. Griveau-Billion et al.

The usual algorithm to find long-only risk parity portfolios, sequential quadratic programming (SQP) to equalize the total contributions to risk, *i.e.*, minimize

$$\sum_{i < j} (\text{TCTR}_i - \text{TCTR}_j)^2,$$

does not scale.

Use coordinate descent to minimize $\sqrt{w'Vw} - \sum b_i \log w$ (where the b_i are the risk budgets): w_i is the positive solution of

$$\sigma_i w_i^2 + \left(\sum_{j \neq i} w_j \rho_{ij} \sigma_j \right) w_i - b_i \sqrt{w'Vw} = 0.$$

Constrained Kelly portfolios under α -stable laws
N. Wesselhöfft (2016)

Add a CVaR constraint to the growth-optimal strategy, and assume α -stable returns.

Managing diversification
A. Meucci (2010)

The eigendecomposition $E'\Sigma E = \Lambda$ of the variance matrix Σ of asset returns X defines *principal portfolios*, with returns $E'X$. Their contributions to portfolio variance $w'\Sigma w$ are $(E^{-1}w) \odot (E^{-1}w) \odot \text{diag } \Lambda$; dividing by $w'\Sigma w$ gives the *diversification distribution* whose entropy defines the **effective number of uncorrelated bets**.

If market exposure is not under our control, remove the first principal portfolio. If there are more constraints, write them as $Aw = b$ and compute the corresponding **conditional eigenvectors**

$$e_i = \underset{\substack{e \text{ such that} \\ e'e=1 \\ Ae=0 \\ \forall j < i \ e'\Sigma e_j=0}}{\text{Argmax}} e'\Sigma e.$$

They can be computed (using Lagrange multipliers) as

$$\begin{aligned} B &= \begin{pmatrix} A \\ e'_1 \Sigma \\ \vdots \\ e'_{j-1} \Sigma \end{pmatrix} \\ P &= I - B'(BB')^{-1}B \\ e_i &= \underset{\|e\|=1}{\text{Argmax}} e'P\Sigma P'e. \end{aligned}$$

We still have $E'\Sigma E = \Lambda$, with Λ diagonal, but $E' \neq E^{-1}$.

One can then look at the mean-entropy frontier.

Building diversified portfolios that outperform out-of-sample
M. López de Prado

Hierarchical risk parity (HRP) aims to circumvent *Markowitz's curse*: the more correlated the investments, the greater the need for diversification, and yet the more likely we will receive unstable solutions. Other approaches include: constraints, Bayesian priors, robust optimization, numerically stable variance estimators (shrinkage, factor models, graphical lasso).

Compute the sample correlation matrix ρ , the corresponding distances $d_{ij} = \sqrt{1 - \rho_{ij}}$ and the distances between those distances $\delta_{ij} = \|d_{\cdot,i} - d_{\cdot,j}\|_2$; perform

single linkage hierarchical clustering and use it to re-order the covariance matrix: it is now near-diagonal.

By analogy with the (unconstrained) minimal variance portfolio, $w = V^{-1}1/1'V^{-1}1$, whose weights are the inverses of the variances when the variance matrix is diagonal:

- Start with the full set of assets $L = \{1, \dots, n\}$ and initial weights $(1, \dots, 1)'$;
- Split L in two, $L = L_1 \cup L_2$ [they use two halves, but with a more balanced hierarchical clustering (e.g., complete linkage), it would make more sense to use it];
- Compute the variances of the corresponding inverse-variance-weighted portfolios

$$V = \begin{pmatrix} V_1 & \\ & V_2 \end{pmatrix}$$

$$w_i \propto (\text{diag } V_1)^{-1}$$

$$v_i = w_i' V w_i;$$

- Rescale the weights [this does not preserve the sum of the weights]

$$w_{I_1} \leftarrow \alpha w_{I_1}$$

$$w_{I_2} \leftarrow (1 - \alpha) w_{I_2}$$

$$\alpha = \frac{v_2}{v_1 + v_2};$$

- Continue recursively.

***Interconnectedness risk
and active portfolio management***
E. Baitinger and J. Papenbrock (2016)

Peripheral stocks, for some measure of centrality (betweenness, closeness, eccentricity, degree, eigenvector) of the minimum spanning tree (or some other “strongly filtered graph”) built from the correlation matrix of daily returns, estimated on a 6-month moving window, have higher risk-adjusted returns. Build a portfolio by minimizing the average centrality (with a few constraints: $w'1 = 1, \|w - w_0\|_1 \leq \varepsilon$ – it is a linear problem); compare with minimum variance, minimum average correlation, maximum diversity, maximum entropy, hierarchical risk parity; measure the performance with Jensen’s alpha, the upside capture ratio

$$UC = \frac{E[\text{portfolio returns} | \text{market up}]}{E[\text{market returns} | \text{market up}]},$$

the downside capture ratio, or the certainty equivalent return for $U(W) = W^{1-\gamma}/(1-\gamma)$ for $\gamma \in \{3, 5, 10\}$.

***Interconnectedness risk
and active portfolio management:
the information-theoretic perspective***
E. Baitinger and J. Papenbrock (2017)

Build minimum centrality portfolios from the *mutual information* rather than the correlation [one could also use the *distance correlation*]. In R check `entropy::mi.plugin` or `infotheo::mutinformation`.

***The real side
of the high-volume return premium***
D. Israeli et al. (2017)

Abnormal volume around earnings announcements increases investor recognition and firm visibility, decreases the cost of capital, and increases future investments. The effect remains after controlling for sales, cash flow, current investments, leverage, Tobin’s Q, earnings surprises, abnormal returns around earnings, etc., but is more pronounced for financially-constrained firms (low assets, low payout ratio) and has been decreasing as information becomes more readily available.

Tobin’s Q is

$$Q = \frac{\text{Market value of assets}}{\text{Book value of assets}}$$

$$= \frac{BV(\text{assets}) + MV(\text{equity}) - BV(\text{equity})}{BV(\text{assets})}$$

***The limits of arbitrage and stock mispricing:
evidence from decomposing
the market to book ratio***
N.M. Al-Shammasi (2017)

Stocks with easily-identified peers have a lower mispricing. The nearest stock is that with the closest *propensity score*, i.e., predicted value of a logistic regression $1_{S\&P\ 500} \sim \text{Fama-French}$. [Other measures of proximity may be more meaningful: density (LOF), correlation, mutual information, distance correlation, cointegration test, etc.]

***Programmed selection
of cyclical turning points***
G. Bry and C. Boschan (1971)

To detect business cycles on a “clean” time series:

- Find local minima and maxima, with a 5-month or 2-quarter window;
- Prune them to ensure there are no consecutive minima or maxima; keep the most extreme;
- Prune them to ensure “up” and “down” periods are at least 5 months long, and the cycles (up-then-down or down-then-up) 15 months;
- Remove extrema too close to or inconsistent with the extremities.

For a real-world time series:

- Apply the procedure to the 12-month moving average;
- Pick the local extrema of the Spencer curve (a moving average with more weight at the center) best matching those points (most extreme in a 5-month radius);
- Pick the local extrema of the raw data best matching those points.

Dating the Euro area business cycle

M. Artis et al.

Markov switching models can help identify business cycles.

The local maxima and minima of a price time series give a set of candidate latent state sequences: we can compute the likelihood of each of them and pick the best. Duration constraints can be imposed by enlarging the state space to contain the (S_1, \dots, S_{15}) satisfying the constraints; the only transitions are $(S_1, \dots, S_{15}) \rightarrow (S_2, \dots, S_{15}, *)$. For monthly data, it may be necessary to filter the time series to remove low frequency patterns, e.g., with a band-pass filter or an HP filter.

Noise fit, estimation error and a Sharpe information criterion

D. Paulsen and J. Söhl (2016)

The realized information ratio tends to be much lower than the forecasted one: correct it.

$$\text{IR}_{\text{out}} = \text{IR}_{\text{in}} - \frac{\# \text{parameters or assets}}{\text{IR}_{\text{in}} \times \# \text{periods}}$$

Option-based benchmark indices – a review of performance and (in)appropriate measures

M. Natter (2017)

Jensen's alpha (the CAPM intercept) is inappropriate to measure the performance of option strategies (for which there are now many indices and even a few ETFs):

- Let the CAPM beta vary with time (e.g., make it a linear or 2-valued function of market returns);
- Add a straddle factor to the CAPM;
- Correct Jensen's alpha for skewness (Leland):

x : market log-returns

y : strategy log-returns

$$b = \frac{E[x]}{\text{Var}[x]}$$

$$\beta = \frac{\text{Cov}(e^x, -e^{-by})}{\text{Cov}(e^x, -e^{-bx})}$$

$$\alpha = E[y] - \beta E[x]$$

- Adjust the Sharpe ratio for higher moments (Stutzer index):

$$I = \text{Max}_{\theta} - \log E[e^{\theta y}]$$

$$\text{IR} = \sqrt{2I} \times \text{sign } E[y].$$

Information geometry and statistical manifolds

M. Suzuki (2014)

Consider a family of probability distributions indexed by an open set $\Omega \subset \mathbf{R}^n$, $S = \{p(\cdot, \xi) : \xi \in \Omega\}$. The Fisher information matrix

$$g_{ij}(\xi) = E[\partial_i \ell_{\xi} \partial_j \ell_{\xi}] = -E[\partial_i \partial_j \ell_{\xi}]$$

where $\ell_{\xi} = \log p(\cdot, \xi)$ and $\partial_i = \partial/\partial \xi_i$, defines a Riemann structure on S . For instance, the Gaussian (1-dimensional) family is the Poincaré half plane

$$\Omega = \mathbf{R} \times \mathbf{R}_+^{\times}$$

$$S = \{N(\mu, \sigma^2) : (\mu, \sigma) \in \Omega\}$$

$$g_{ij}(\mu, \sigma) = \begin{pmatrix} \sigma^{-2} & 0 \\ 0 & 2\sigma^{-2} \end{pmatrix}$$

The α -connection,

$$\begin{aligned} \Gamma_{ijk}^{\alpha} &= \langle \nabla_{\partial_i}^{\alpha} \partial_j, \partial_k \rangle \\ &= E \left[\left(\partial_i \partial_j \ell_{\xi} + \frac{1-\alpha}{2} \partial_i \ell_{\xi} \partial_j \ell_{\xi} \right) \partial_k \ell_{\xi} \right], \end{aligned}$$

for $\alpha \in \{-1, 0, +1\}$, satisfies

$$\Gamma_{ijk}^{\beta} = \Gamma_{ijk}^{\alpha} + \frac{\alpha - \beta}{2} T_{ijk}$$

$$T_{ijk} = E[\partial_i \ell_{\xi} \partial_j \ell_{\xi} \partial_k \ell_{\xi}]$$

$$\nabla^{\alpha} = (1 - \alpha) \nabla^0 + \alpha \nabla^1 = \frac{1 + \alpha}{2} \nabla^1 + \frac{1 - \alpha}{2} \nabla^{-1}$$

where

- ∇^0 is the Levi-Civita connection (compatible with the metric, torsion-free);
- ∇^1 is called the *exponential connection*; it is flat if S is an exponential family;
- ∇^{-1} is called the *mixture connection*; it is flat if S is a mixture family.

Computations with ∇^1 and ∇^{-1} are often easier than those with ∇^0 : they involve divergences rather than geodesic distances.

Information geometry for neural networks

D. Wagenaar (1998)

Families of distributions are often parametrized by an open of \mathbf{R}^n : this endows them with a geometric structure (differential manifold), but the Euclidean distance is inappropriate. Fisher information gives a better (parametrisation-independent) Riemannian structure:

$$g_{\mu\nu} = E[\partial_{\mu} \ell \partial_{\nu} \ell] = -E[\partial_{\mu} \partial_{\nu} \ell].$$

A *parallel transport* is a differentiable way of comparing tangent spaces at different points, when they are linked by a curve,

$$\Phi_{P \rightarrow P'} : T_{P'} M \longrightarrow T_P M,$$

with $\Phi_{P \rightarrow P} = \text{Id}$. Its first Taylor expansion, for $P = \gamma(0)$, $P' = \gamma(\varepsilon)$, $\dot{\gamma}(0) = \partial_{\mu}$, can be written

$$\Phi_{P \rightarrow P'}(\partial_{\nu}) = \partial_{\nu} + \varepsilon \Gamma_{\mu\nu}^{\rho} \partial_{\rho} + o(\varepsilon)$$

and defines a *connection* Γ .

Tangent vectors have several interpretations:

- 1-jets, i.e., equivalence classes of curves;
- Derivatives, ∂_{μ} ;

– Random variables, $\partial_\mu \ell$, *i.e.*, functions of X .

This last interpretation is specific to statistical manifolds and allows us to define two new connections. A connection is a mapping between $T_\theta M$ and $T_{\theta+\delta\theta}M$, whose bases are

$$\begin{aligned}\partial_\mu \ell(\theta) \\ \partial_\mu \ell(\theta + \delta\theta) = \partial_\mu \ell(\theta) + \partial_\mu \partial_\nu \ell(\theta) \delta\theta^\nu + O(\delta\theta^\mu \delta\theta^\nu).\end{aligned}$$

We can modify the second term to make its expectation disappear (this looks desirable since $E[\partial_\mu \ell(\theta)] = 0$) in two ways

$$\begin{aligned}\partial_\mu \partial_\nu \ell(\theta) + E[\partial_\mu \ell \partial_\nu \ell] \\ \text{or } \partial_\mu \partial_\nu \ell(\theta) - \partial_\mu \ell \partial_\nu \ell.\end{aligned}$$

Projecting the random variable onto $T_\theta M$,

$$\text{proj} : A \mapsto E[A \partial_\nu \ell] g^{\mu\nu} \partial_\mu \ell$$

gives

$$\begin{aligned}\Gamma_{\mu\nu}^\lambda &= E[\partial_\mu \partial_\nu \ell \partial_\rho \ell] g^{\rho\lambda} \\ \text{and } \Gamma_{\mu\nu}^\lambda &= E[\partial_\mu \partial_\nu \ell \partial_\rho \ell + \partial_\mu \ell \partial_\nu \ell \partial_\rho \ell] g^{\rho\lambda}.\end{aligned}$$

This defines the family of α -connections

$$\Gamma_{\mu\nu}^{(\alpha)\lambda} = E\left[\partial_\mu \partial_\nu \ell \partial_\rho \ell + \frac{\alpha-1}{2} \partial_\mu \ell \partial_\nu \ell \partial_\rho \ell\right] g^{\rho\lambda}.$$

The metric connection

$$\Gamma_{\mu\nu}^\lambda = \frac{1}{2} [\partial_\mu g_{\nu\rho} + \partial_\nu g_{\mu\rho} - \partial_\rho g_{\mu\nu}] g^{\rho\lambda}$$

is $\Gamma^{(0)}$.

Parallel transport does not usually preserve scalar products. Two connections Γ and Γ^* are *dual* if the vector fields X and Y^* , defined as the parallel transport of $X|_{\gamma(0)}$ and $Y|_{\gamma(0)}$ wrt Γ and Γ^* along a curve γ satisfy

$$\forall t \quad \langle X(t), Y^*(t) \rangle = \langle X(0), Y(0) \rangle.$$

The α and $-\alpha$ connections are dual.

For dually flat connections, Γ , Γ^* , there exist local coordinate systems θ , $\tilde{\theta}$, such that $\Gamma \equiv 0$ and $\Gamma^* \equiv 0$, and potential functions $\Theta(\theta)$, $\tilde{\Theta}(\tilde{\theta})$ such that $\tilde{\theta}_\mu = \partial_\mu \Theta(\theta)$ (and conversely). The *divergence* between two points P and Q is

$$D(P, Q) = \Theta(\theta_P) + \tilde{\Theta}(\tilde{\theta}_Q) - \theta_P^\mu \tilde{\theta}_{Q,\mu}.$$

It behaves like a distance, but is easier to compute: there is no need to integrate along a geodesic.

$$\begin{aligned}D(P, Q) &\geq 0 \\ D(P, Q) = 0 &\equiv P = Q \\ \left. \frac{\partial D(P, Q)}{\partial \theta} \right|_{P=Q} &= 0 \\ \frac{\partial^2 D(P, Q)}{\partial \theta^\mu \partial \theta^\nu} &= g_{\mu\nu}(P)\end{aligned}$$

A computational approach to Fisher information geometry with applications to image analysis W. Mio et al.

Given two probability density functions (pdf) on $[0, 1]$, how can we deform one into the other? Those pdfs form an infinite dimensional statistical manifold (we can represent a pdf p with its log-likelihood $\phi = \log p$):

$$\begin{aligned}\mathcal{P} &= \{ \phi : \int e^\phi = 1 \} \\ T_\phi \mathcal{P} &= \{ f : \int f e^\phi = 0 \} \\ \langle f, g \rangle &= \int f g e^\phi\end{aligned}$$

(for the scalar product, use the Hessian of the Kulback-Leibler divergence). Discretizing the pdf on an equispaced grid gives

$$\begin{aligned}\mathcal{P}_n &= F^{-1}(1) \\ F : \begin{cases} \mathbf{R}^n & \longrightarrow \mathbf{R} \\ \phi & \longmapsto \sum e^{\phi_i} \end{cases} \\ \langle f, g \rangle_\phi &= \sum f_i g_i e^{\phi_i} \\ T_\phi \mathcal{P}_n &= \{ f : \langle f, \mathbf{1} \rangle_\phi = 0 \} \\ dF_\phi(f) &= \sum f_i e^{\phi_i} = \langle f, \mathbf{1} \rangle_\phi\end{aligned}$$

Geodesics in a given direction can be computed by iterating the following steps (?).

$$\begin{aligned}\phi_0 &: \text{starting point} \\ f_0 &: \text{direction} \\ \phi_1 &\leftarrow \phi_0 + \varepsilon f \\ \phi_1 &\leftarrow \phi_1 + \alpha \mathbf{1}, \text{ with } \alpha \text{ s.t. } F\phi_1 = 1 \\ f_1 &\leftarrow f_0 - \langle f_0, \mathbf{1} \rangle_{\phi_1} \mathbf{1} \\ f_1 &\leftarrow \|f_0\| \frac{f_1}{\|f_1\|}\end{aligned}$$

For the geodesic between two points, use the shooting method to find the correct direction to reach the desired target in unit time.

Those geodesics can help compute Fréchet means, perform PCA or k -means.

Computational information geometry for machine learning F. Nielsen (MLSS 2015)

1. A statistic t is *sufficient* for a parameter θ if

$$P(X | t(X), \theta) = P(X | t(X)).$$

The probability can then be decomposed as (Fisher–Neyman theorem)

$$P(x; \theta) = g(t(x), \theta) h(x).$$

This motivates the **exponential family**:

$$p(x; \theta) = \exp[\langle t(x), \theta \rangle - F(\theta) + k(x)]$$

$t(x)$: sufficient statistic
 $k(x)$: auxiliary term
 $F(\theta)$: normalization constant.

Many common distributions are exponential (Uniform, Cauchy and stable distributions are not exponential).

The normalizing constant

$$F(\theta) = \log \int \exp[\langle t(x), \theta \rangle + k(x)] dx$$

is convex. Its convex conjugate is

$$F^*(\eta) = \sup_{\theta \in \Theta} \langle \eta, \theta \rangle - F(\theta),$$

for $\eta \in H$, $H = \{\nabla F(\theta) : \theta \in \Theta\}$, and the supremum is obtained for $\theta = \nabla F^*(\eta) = (\nabla F)^{-1}(\eta)$. The *canonical divergence* is (Fenchel–Young inequality)

$$A_F(\theta : \eta) = F(\theta) + F^*(\eta) - \langle \eta, \theta \rangle \geq 0.$$

Exponential families have two parametrizations.

$$\begin{array}{ccc} \text{natural} & & \text{expectation} \\ \theta \in \Theta & \longleftrightarrow & \eta \in H \\ \theta = \nabla F^*(\eta) & & \eta = \nabla F(\theta) = E[t(X)]. \end{array}$$

The maximum likelihood estimator is easy to compute in expectation coordinates: $\eta = E[t(X)]$.

2. The Fisher information is the variance of the *score*, i.e., the information a random variable X carries about a parameter θ . It defines a metric on statistical manifolds

$$g_{ij}(\theta) = E[\partial_i \ell \partial_j \ell] = -E[\partial_i \partial_j \ell] = 4 \int \partial_i \sqrt{p} \partial_j \sqrt{p} dx.$$

The *exponential map* maps (a star-shaped neighbourhood of the origin in) $T_x M$ to M : $v \mapsto \exp_x(v) = \gamma_v(1)$; its inverse is \log_x . Location-scale families (Gaussian, Cauchy, Student, etc.) are hyperbolic (they have negative curvature); the multinomial family is spherical.

On a statistical manifold, one can interpret tangent vectors as random variables, e.g., $\partial_i p(x)$, $\partial_i \ell(x)$, $2\partial_i \sqrt{p(x)}$, or, more generally, $\partial_i f_\alpha(x)$, where

$$f_\alpha(u) = \begin{cases} \frac{2}{1-\alpha} u^{\frac{1-\alpha}{2}} & \alpha \neq 1 \\ \log u & \alpha = 1 \end{cases}$$

For a univariate Gaussian, the Fisher information is

$$I(\mu, \sigma) = \sigma^{-2} \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix};$$

after rescaling, this is the Poincaré half plane. Positive matrices are also hyperbolic spaces, e.g., the Poincaré disk (conformal) or the Klein disk (not conformal).

Information geometric optimization uses the natural gradient $I^{-1} \nabla f$ instead of the gradient ∇f .

Jeffrey's prior, $q(\theta) \propto \sqrt{|g(\theta)|}$, is invariant under reparametrization.

3. The Riemannian metric is not the only way of defining geodesics and “distances”. Given two probability distributions p and q , one can consider curves between them:

$$\begin{aligned} \gamma_m : \quad & r(x, t) = \alpha p(x) + (1 - \alpha) q(x) \\ \gamma_e : \quad & \log r(x, t) = \alpha \log p(x) + (1 - \alpha) \log q(x) + c. \end{aligned}$$

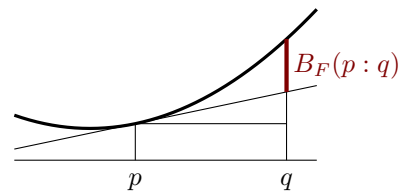
They are geodesics for the dual *mixture* (m) and *exponential* (e) **connections**. For the exponential family, they are flat, and $\Gamma \equiv 0$, resp. $\Gamma^* \equiv 0$, in the $\theta = \nabla F^*(\eta)$ and $\eta = \nabla F(\theta)$ coordinate. The metric tensor can be recovered from the *potential functions* F and F^* :

$$\begin{aligned} g_{ij}(\theta) &= \frac{\partial^2 F(\theta)}{\partial \theta_i \partial \theta_j} & \Gamma_{ijk}^\alpha &= \frac{1 - \alpha}{2} \frac{\partial^3 F(\theta)}{\partial \theta_i \partial \theta_j \partial \theta_k} \\ g^{ij}(\eta) &= \frac{\partial^2 F^*(\eta)}{\partial \eta_i \partial \eta_j} & \kappa &= \frac{1 - \alpha^2}{4} \end{aligned}$$

If $p(x; \theta) = \exp[\langle \theta, x \rangle - F(\theta)]$, F is the *cummulant* dunction and F^* the *negentropy*. Since $\nabla^{(e)} = \nabla^{(1)}$ and $\nabla^{(m)} = \nabla^{(-1)}$ are flat, the geodesics are easier to compute than those for the Levi-Civita connection $\nabla^{(0)}$.

4. If P and Q belong to the same exponential family, the canonical and Bregman **divergences** are related.

$$\begin{aligned} A_F(\theta : \eta) &= F\theta + F^*\eta - \langle \eta, \theta \rangle \\ B_F(\theta_1 : \theta_2) &= F\theta_1 - F\theta_2 - \langle \theta_1 - \theta_2, \nabla F\theta_2 \rangle \\ KL(P \parallel Q) &= B_F(\theta_P : \theta_Q) = B_{F^*}(\eta_P : \eta_Q) \\ &= A_F(\theta_Q : \eta_P) = A_{F^*}(\eta_P : \theta_Q) \end{aligned}$$



(The Bregman divergence is the residual of the first order Taylor expansion.)

5. The **maximum entropy** distribution p with prescribed moments $E[t(X)]$ minimizes the Kullback-Leibler divergence with the uniform distribution $q = (\frac{1}{n}, \dots, \frac{1}{n})$.

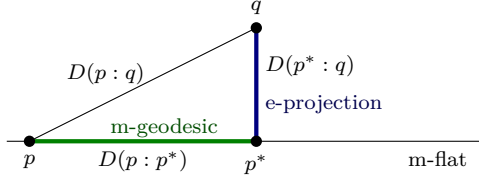
$$\begin{aligned} \text{Find} \quad & p \\ \text{To maximize} \quad & D(p : q) = \sum p_i \log \frac{p_i}{q_i} \\ \text{Such that} \quad & \forall j \sum_i p_i t_{ij} = m_j \\ & \sum_i p_i = 1 \\ & \forall i \ p_i \geq 0 \end{aligned}$$

Using Lagrange multipliers, we see that this distribution is exponential $p(x) \propto q(x) \exp\langle \theta, t(x) \rangle$.

The KL-Pythagoras theorem,

$$KL(p : q) = KL(p : p^*) + KL(p^* : q)$$

holds if p and p^* satisfy the moment constraints, the submanifold defined by those constraints is m-flat, and p^* is the e-projection of q on it.



6. The maximum entropy estimator is the e-projection of a prior onto the moment manifold.

The maximum likelihood estimator is the m-projection of the empirical distribution p_{emp} onto the model submanifold M ,

$$p_\theta = \underset{p \in M}{\text{Argmin}} D(p_{\text{emp}} : p).$$

Those **information projections** are defined as

$$\text{e-projection}_M(q) = \underset{p \in M}{\text{Argmin}} D(p : q)$$

$$\text{m-projection}_M(q) = \underset{p \in M}{\text{Argmin}} D(q : p)$$

The e- (resp. m-) projection is unique if M is m- (resp. e-) flat.

7. The Riemannian distance (*Rao distance*) is difficult to compute explicitly (it is not known in closed form for multivariate normals), but lets you use the exponential (and logarithm) map between the tangent space and the manifold.

One often switches between conformal models (e.g., the Poincaré disk, good for visualization) and non-conformal ones (e.g., the Klein disk, whose geodesics are straight lines – good for computations).

8. Many geometric notions can be generalized to divergences.

The cosine law ($c^2 = a^2 + b^2 - 2ab \cos \hat{C}$) becomes

$$\begin{aligned} D(P : R) &= D(P : Q) + D(Q : R) - \|\dot{\gamma}_{PQ}\| \|\dot{\gamma}_{QR}^*\| \cos \theta \\ &= D(P : Q) + D(Q : R) - \langle \theta_P - \theta_Q, \eta_R - \eta_Q \rangle \end{aligned}$$

where θ is the angle between the ∇ -geodesic γ_{PQ} and the ∇^* -geodesic γ_{QR} .

The *Bregman sphere* is defined as follows.

$$F : \mathbf{R}^d \longrightarrow \mathbf{R}$$

$$\mathcal{F} = \{ (x, F(x)) : x \in \mathbf{R}^d \} \quad \text{hypersurface}$$

$$H_p(x) = \langle x - p, \nabla F_p \rangle + F_p \quad x, p \in \mathbf{R}^p$$

$$\mathcal{H}_p = \{ (x, H_p(x)) : x \in \mathbf{R}^d \} \quad \text{tangent space } T_{(p, F_p)} \mathcal{F}$$

$$F \cap (\mathcal{H}_p + (\mathbf{0}, r)) \quad \text{Bregman sphere}$$

Vantage point trees partition the space with Bregman spheres to speed up nearest neighbour queries. One can (algorithmically) compute the smallest Bregman enclosing ball, test if a point is inside a Bregman ball defined by $n + 1$ support points (to compute Delaunay triangulations), etc.

9. There are many divergences

$$\text{Csiszár} \quad I_f(p, q) = \sum q_i f\left(\frac{p_i}{q_i}\right)$$

$$\text{KL} \quad KL(p : q) = \sum p_i \log \frac{p_i}{q_i}$$

$$\text{Bregman} \quad B_F(x, y) = Fx - Fy - \langle x - y, \nabla Fy \rangle$$

$$\text{Canonical} \quad A_F(\theta : \eta) = F\theta + F^*\eta - \langle \eta, \theta \rangle$$

$$\begin{aligned} \text{Skew-Jensen} \quad J_\alpha(x, y) &= \alpha Fx + (1 - \alpha)Fy \\ &\quad - F(\alpha x + (1 - \alpha)y) \end{aligned}$$

and many divergence-based k -means generalizations.

Multiclass classification

Y. Keshet (2014)

The all-against-one and all-pairs multiclass classification schemes reduce the problem to binary classification

$$\underset{i}{\text{Argmax}} P(X = i)$$

$$\text{or } \underset{i}{\text{Argmax}} \sum_{j \neq i} P(X = i \mid X \in \{i, j\});$$

they can be generalized to an arbitrary set of binary comparisons $P(X \in I \mid X \in J)$, $I \subsetneq J$ with error correcting codes (ECOC: error correction output codes).

Multiclass classifiers (softmax, SVM, etc.) are usually of the form $\underset{i}{\text{Argmax}} \text{Score}(X, i)$. (the formula is the same, but there is only one model).

Multi-label classification with error-correcting codes

C.S. Ferng and H.T. Lin

Error correcting codes (ECOC) can also be used for multi-label classification (predicting a *set* of labels instead of just one).

Here are the common error correcting codes.

- Repetition: repeat each bit 3 times.
- Hamming(7,4): 4 data bits and 3 parity bits.

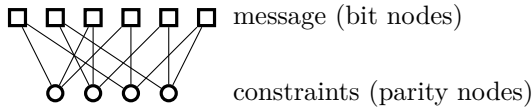
$$p_1 = d_1 \oplus d_2 \oplus d_4$$

$$p_2 = d_1 \oplus d_3 \oplus d_4$$

$$p_3 = d_2 \oplus d_3 \oplus d_4$$

- Low-density parity check (LDPC) codes do not really specify how to encode a message (this is left as an exercise for the user) but rather which messages are valid. A binary vector x is valid if $Px = 0$, where P is the binary matrix (parity matrix) defining the code. It usually has the same number of 1s in each

row, and the same number of 1s in each column. It can be represented as a bipartite graph.



One can decode a message (*i.e.*, find the closest valid message) using message passing.

- BCH are the most commonly-used codes.

$q = 2$ prime number
 $n = q^m - 1$
 $d < n$
 $\alpha \in \mathbf{F}_{q^m}$ generator of \mathbf{F}_{q^m}
 $m_i \in \mathbf{F}_q[X]$ minimal polynomial of α^i
 $g = \text{lcm}(m_1, \dots, m_d)$
 Code words = $\{ P \in \mathbf{F}_q[X] : \deg P < n \text{ and } g|P \}$

**Stochastic portfolio theory:
a machine learning perspective
Y.L.K. Samo and A. Vervuurt**

Stochastic portfolio theory shows that, under reasonable assumptions, a (continuously-rebalanced) portfolio with weights $w_i \propto \text{MCap}_i^p$ (diversity-weighted portfolio), for $p \in (0, 1)$, a.s. outperforms the market. The amplitude of the performance and the time needed to reach it depend on p : small, quick out-performance if p is close to 1, larger, but more delayed if p is close to 0 (the $1/N$ portfolio).

To find the best p , empirically, *i.e.*, that maximizing the excess returns (after 10bp transaction costs), one can use a grid search ($p \in [-8, 8]$) or a Bayesian approach (Metropolis-Hastings), with a uniform prior on p , and $\text{Gamma}(\text{ExcessReturns}(p); a, b)$ as likelihood, where a and b reflect the investor's risk appetite (e.g., a bullish investor could want mean= 7 and sd= .5).

A non-parametric approach,

$$w_i \propto f(\text{MCap}_i)$$

$$\log f \sim \text{GP}(0, k),$$

where k is a rational quadratic kernel,

$$k(x, y) = a \left[1 + \frac{(x - y)^2}{b} \right]^c,$$

prefers midcaps, and can be generalized to account for other predictors, e.g., the ROA,

$$w_i \propto f(\text{MCap}_i, \text{ROA}_i)$$

$$\log f \sim \text{GP}(0, k_{\text{MCap}} \times k_{\text{ROA}}).$$

**Diversity-weighted portfolios
with negative parameter
A. Vervuurt and I. Karatzas (2015)**

Diversity-weighted portfolios $w_i \propto \text{MCap}_i^p$, with $p < 0$, are also worth considering.

**Bayesian Lipschitz constant
estimation and quadrature
J.P. Calliess**

Probabilistic numerics provides quadrature rules tailored to a given prior: for instance, the trapezoidal rule comes from a random walk prior and is suboptimal if we know the function is smoother.

**What works best when? A framework
for systematic heuristic evaluation
I. Dunning et al. (2015)**

Use machine learning to select which heuristic (or machine learning algorithm) to use on a new problem (MaxCut or quadratic unconstrained binary optimization).

$$\text{MaxCut:} \quad \text{Maximize} \sum_{i,j} w_{ij} (1 - y_i y_j)$$

$$\text{QUBO:} \quad \text{Maximize} x' Q x$$

Train on real (non-simulated) data, with the following graph metrics as features: number of nodes, number of edges, assortativity, approximate maximum independent set, Laplacian eigenvalues, approximate chromatic number; mean, standard deviation, minimum, maximum, skew, kurtosis of degree, weight, average neighbour degree (as a function of the node degree), clustering coefficient, core number.

**Obtaining calibrated probability estimates from
decision trees and naive Bayesian classifiers
B. Zadrozny and C. Elkam**

Machine learning classifiers may provide accurate class forecasts and rankings, but rarely accurate probability estimates.

- Since predictors, in a naive Bayes classifier, are often correlated, the probabilities are close to 0 or 1: a histogram method can rescale them.
- Decision trees try to make leaves homogeneous: observed probabilities are shifted towards 0 or 1; probabilities in small leaves have high variance. *Laplace smoothing* replaces probabilities k/n with $(k+1)/(n+2)$, *i.e.*, shrinks them towards $1/2$ – but this need not be the base rate. Instead, shrink them towards the base rate b , $k/n \mapsto (k+bm)/(n+m)$ (*m-estimation*), with $bm = 10$, or towards the parent node. Splitting criteria generating smaller trees (e.g., $h(p, 1-p) = \sqrt{p(1-p)}$) may help.

**Scalable inference
for structured Gaussian process models
Y. Saatçi (2011)**

A structured Gaussian process (GP) is a GP whose covariance matrices have a special structure, often allowing more efficient computations:

- Bounded rank, as in linear regression;

- Toeplitz, for 1-dimensional kernels evaluated on a regular grid;
- Gauss-Markov processes, for some 1-dimensional kernels (e.g., Matérn);
- Block structure, if there is a change point;
- Additivity, as in GAMs;
- Kronecker product, if the kernel is a tensor product evaluated on a (not necessarily regular) grid).

Generalized GPs (i.e., non-Gaussian “GP”s, e.g., Poisson GPs for count data) can be estimated using the Laplace approximation or expectation maximization.

1. A linear stochastic differential equation (SDE) with constant coefficients

$$f^{(m)}(x) + a_{m-1}f^{(m-1)}(x) + \dots + a_0f(x) = W(x)$$

defines a GP with the Markov property

$$\frac{d}{dx} \begin{pmatrix} f \\ f' \\ \vdots \\ f^{(m-1)} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & \vdots \\ -a_0 & \dots & \dots & \dots & -a_m \end{pmatrix} \begin{pmatrix} f \\ f' \\ \vdots \\ f^{(m-1)} \end{pmatrix} + \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} W;$$

it is a state space model: it can be estimated with a Kalman smoother (after sorting the data along x). The SDE can be seen as a *linear time-invariant* (LTI) system, $f = Tw$, $w = \text{noise}$. It is characterized by its impulse function $h = T\delta$:

$$P\phi = \phi * h \\ k(\tau) = q\delta(\tau) * h(\tau) * h(-\tau).$$

Starting with

$$k(\tau) = q\delta(\tau) * h(\tau) * h(-\tau) \\ Tw = w * h$$

$$\sum a_k f^{(k)}(x) = w(x)$$

and taking the Fourier transform

$$S(\omega) = q |H(\omega)|^2 \\ F(\omega) = W(\omega)H(\omega) \\ \sum a_k (i\omega)^k F(\omega) = W(\omega)$$

gives

$$S(\omega) = q |H(\omega)|^2 = a \left| \frac{F(\omega)}{W(\omega)} \right|^2 \\ = \frac{1}{|\sum a_k (i\omega)^k|^2} \\ = \frac{1}{|\text{polynomial in } i\omega|^2} \\ = \frac{1}{\text{polynomial in } \omega^2}.$$

For the Matérn family,

$$S_\nu(\omega) \propto \frac{1}{(\lambda^2 + \omega^2)^{\nu+1/2}}.$$

For instance, $\nu = 7/2$ gives

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\lambda^4 & -4\lambda^3 & -6\lambda^2 & -4\lambda \end{pmatrix}$$

(this also explains why the paths of a Matérn GP are a.s. differentiable $\nu - 1/2$ times).

For the square exponential kernel, $S(\omega) \propto \exp(-\alpha\omega^2)$ is not the inverse of a polynomial, but can be approximated using polynomials (it may be easier to use a Matérn kernel, though).

Splines can also be expressed as GPs.

2. The Bayesian online change point detection (BOCPD) algorithm combines a base model and a hazard function; the parameters of the Bayesian model change at each changepoint.

r_t	length of the current run
$y_{t-r}, \dots, y_{t-1}, y_t$	current run
$P[y_t y_{t-r}, \dots, y_{t-1}]$	base model

The posterior $P[r_t | y_1, \dots, y_{t-1}]$ can be used to compute predictions robust to regime changes (with a message passing algorithm). For Gaussian processes, this is untractable, but it can be approximated with a grid or HMC; in both cases, prune highly unlikely run length values.

3. Additive GPs are GAMs with GP components.

4. The covariance matrices of a GP with a product kernel on a (not necessarily regular) grid are Kronecker products and therefore amenable to efficient computations.xs

Bandit algorithms for searching large spaces **L.R.M. Dorard (2012)**

A bandit game is a 1-state Markov decision process (MDP).

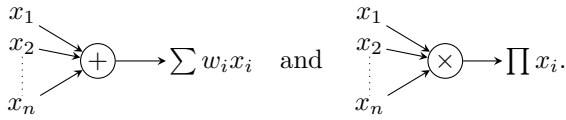
The *many-arm bandit* problem has too many arms (actions) to explore them all, but they are often arranged in a tree: one can use a Gaussian process to model the reward of a leaf, from the features of the path from the root (explicit features are not needed – a kernel suffices, e.g., that given by the number of nodes in common).

Both bandit problems and Bayesian optimization are exploration/exploitation problems.

Sum-product networks: a new deep architecture **H. Poon and P. Domingos**

The *network polynomial* of an (unnormalized) probability distribution P on $\{0, 1\}^n$ is $\sum_{I \subseteq [1, n]} P[x_I = 1, x_{\bar{I}} = 0] x_I x_{\bar{I}} \in \mathbf{R}[x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n]$. It can be represented as a *sum-product network*, i.e., a network

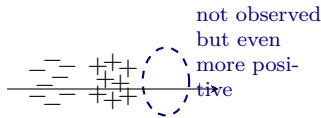
whose inputs are $x_1, \dots, x_n, \bar{x}_1, \dots, \bar{x}_n$ and with two types of nodes,



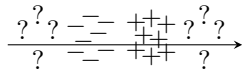
(Not all sum-product networks represent network polynomials, but there are simple sufficient conditions to ensure it.)

Adversarial perturbations of deep neural networks D. Warde-Farley et al.

Deep neural networks are affected by adversarial examples: one can take a correctly classified input and modify it slightly to produce a different output. Worryingly, adversarial examples seem model-agnostic. The problem is not with the excessive non-linearity of neural nets but their insufficient non-linearity: instead of a linear situation



we would prefer



Generative adversarial networks (GAN) simultaneously learn a generator, which transforms noise into an adversarial example, and a discriminator, whose role is to tell if the input came from the data or the generator – a minimax game.

A consistent multivariate test of association based on ranks of distances R. Heller et al. (2012)

To test the independence of two quantitative random variables X and Y , a permutation test for

$$\sum_{i \neq j} \chi^2 [X \in B(x_i, d_{ij}^X) \perp\!\!\!\perp Y \in B(y_i, d_{ij}^Y)]$$

is an alternative to the distance correlation (`energy::dcov.test`).

Stochastic gradient Riemannian Langevin dynamics on the probability simplex S. Patterson and Y.W. Teh

Langevin dynamics,

$$d\theta_t = \nabla \log \pi(\theta) dt + 2\sigma dW_y,$$

i.e., gradient updates plus noise

$$\begin{aligned} \theta_{n+1} &\leftarrow \theta_n + \varepsilon [\nabla_{\theta} \log p(\theta) + \sum \nabla_{\theta} \log p(x_i|\theta)] + \zeta \\ \zeta &\sim N(0, \varepsilon) \end{aligned}$$

(you can add a Metropolis-Hastings (MH) correction for the discretization, *i.e.*, only accept the proposal with the MH probability) can be preconditioned by using the Fisher information Riemannian structure (or some approximation, if it is intractable) and used on mini-batches.

Among the many parametrizations of the probability simplex,

$$\begin{aligned} \pi_n &= 1 - \sum_1^{n-1} \theta_i \\ \pi_k &= \frac{e^{\theta_k}}{1 + \sum_1^{n-1} e^{\theta_i}} \\ \pi_k &= \frac{\theta_k}{\sum \theta_i} \\ \pi_k &= \frac{e^{\theta_k}}{\sum e^{\theta_i}} \end{aligned}$$

the expanded mean $\pi_k = |\theta_k| / \sum |\theta_i|$, $G(\theta) = \text{diag } \theta^{-1}$ (Fisher information from a Gamma-Poisson model – the parametrization is redundant) works better.

MCMC for continuous-time discrete-state systems V. Rao and Y.W. Teh

A *semi-Markov (jump) process* (sMJP) on N states is defined by an $N \times N$ matrix of hazard functions describing the state transitions. When the hazard functions are constant, it is a Markov jump process. To sample from a sMJP, proceed as for non-homogeneous Poisson processes.

Temporal disaggregation of time series C. Sax and P. Steiner (2013)

The `tempdisagg` package disaggregates time series, *i.e.*, converts annual time series to monthly (or quarterly) ones, using auxiliary monthly series (countries providing quarterly GDP estimates interpolate annual numbers in this way). It is a 2-step process: first, interpolate the series using GLS and assuming it is $AR(p)$; then, tweak it, so that the sum, average, first or last value of the monthly series be consistent with the annual one.

Compressive spectral clustering N. Tremblay et al.

Spectral clustering on large graphs using subsampling (of matrix columns), random sampling (of graph signals, *i.e.*, functions defined on the (finite) set of nodes) or iterative node aggregation.

***A general framework
for updating belief distributions***
P.G. Bissiri et al. (2013)

In Bayesian statistics, the belief update

$$\begin{aligned}\text{Posterior}(\theta) &\propto \text{Prior}(\theta) \cdot \text{Likelihood}(\text{Data}|\theta) \\ &\propto \text{Prior}(\theta) \cdot \exp -\text{loss}(\text{Data}; \theta)\end{aligned}$$

still makes sense for an arbitrary loss function (e.g., from robust statistics, instead of minus the log-likelihood), which can have much fewer parameters than a full model and is more robust to “model” misspecification. This can be theoretically justified by

$$\text{Posterior}(\theta) = \underset{\nu}{\text{Argmin}} \text{KL}(\nu \parallel \text{prior}) + \int_{\Theta} \text{loss}(\text{Data}; \theta) \nu(d\theta).$$

***A survey on independence-based
Markov networks learning***
F. Schlüter (2013)

Algorithms to learn the structure of a Markov network are

- Either score-based: start with 1-variable potentials; add merged potentials, if they improve the model enough; iterate;
- Or independence-based: compute the Markov blanket (MB) of each node with independence tests and join them (OR rule):
 - The grow-shrink (GS) algorithm adds the variables needed to the MB, and then removes those no longer needed;
 - The incremental association MB (IAMB) algorithm is similar, but adds the best variables first;
 - The GS inference Markov network (GSIMN) algorithm uses the triangle theorem to reduce the number of tests;
 - etc. (prefer IAMB variants or IPC-MB).

***Using Markov blankets
for causal structure learning***
J.P. Pellet and A. Elisseff (2008)

A random variable X_i is *strongly relevant*, *weakly relevant* or *irrelevant* to a target Y if

- (i) $P(Y|X_{\setminus i}) \neq P(Y|X_{\setminus i}, X_i)$
- (ii) $\exists S \subset X_{\setminus i} \quad P(Y|S) \neq P(Y|S, X_i)$
- (iii) $\forall S \subset X_{\setminus i} \quad P(Y|S) = P(Y|S, X_i)$

In a faithful Bayesian network, the **Markov blanket** of a node (*i.e.*, the children, parents and spouses (children’s parents), *i.e.*, the neighbours in the moral graph) is the set of strongly relevant variables; feature selection algorithms can estimate them. It is possible to recover the Bayesian network up to *observational equivalence* (*i.e.*, (unoriented) adjacencies and (oriented) V-structures) by looking for spouse links in triangles (*i.e.*, finding d-separation sets), removing them, and orienting the remaining two edges.

***Numeric experiments
on the commercial quantum computer***
R.H. Warren (2013)

Adiabatic quantum computers (D-wave – the only commercial quantum computers available) solve unconstrained binary quadratic optimization problems

$$\underset{s}{\text{Argmin}} s'Js + h's.$$

Many combinatorial optimization problems can be cast in this form.

This looks unrelated to textbook quantum computers, which do linear algebra in $(\mathbb{C}^2)^{\otimes n}$.

***A unified framework for modeling and solving
combinatorial optimization problems:
a tutorial***
G.A. Kochenberger and F. Glover (2004)

Many combinatorial optimization problems can be reformulated as *unconstrained* quadratic binary programs. Most constraints can be recast as penalties, e.g.,

$$\begin{aligned}x + y \leq 1 &\rightsquigarrow Pxy \\ x + y \geq 1 &\rightsquigarrow P(1 - x - y + xy) \\ x + y = 1 &\rightsquigarrow P(1 - x - y + 2xy) \\ x \leq y &\rightsquigarrow P(x - xy) \\ \sum x_i \leq 1 &\rightsquigarrow P \sum_{i < j} x_i x_j\end{aligned}$$

with P sufficiently large.

***Solving the optimal trading trajectory problem
using a quantum annealer***
G. Rosenberg et al.

Small multi-period portfolio optimization problems can be solved with a quantum computer (50 assets, 10 periods already require more than the 1152 bits D-Wave offers).

***Quantum algorithms for supervised
and unsupervised machine learning***
S. Lloyd et al. (2013)

When data is stored in a qRAM (quantum RAM), computing distances and inner products (e.g., for k-means) of vectors in N -dimensional space takes time $O(\log N)$.

Quantum machine learning enhances privacy: the user only accesses an exponentially small fraction of the data.

***Predicting the present with Bayesian
structural time series***
S.L. Scott and H. van de Veerdijk (2013)

Nowcasting models a time series from its past values and from contemporaneous easy-to-measure time series, $y_t \sim y_{t-1} + x_t$. Combine a structural model for the time series with spike-and-slab priors for the regression coefficients.

Cuckoo filter: practically better than Bloom
B. Fan et al. (2014)

A **cuckoo hash table** has two hash functions, h_1 and h_2 , and stores an element x , either in $h_1(x)$ or $h_2(x)$ – if both are occupied, it displaces one of its occupants, which moves to its other location, possibly displacing other elements – if this fails after 500 iterations, the hash table is full.

They can be adapted to work as Bloom filters with deletion (for the same space usage as a Bloom filter).

***Model-based variable decorrelation
in linear regression***
C. Théry

To estimate a regression in the presence of correlated variables, one can use

- Shrinkage (ridge, lasso), variable selection (spike-and-slab);
- Clusters of variables with the same coefficients

or model the correlation structure, e.g., with a sparse set of regressions between the predictors – the **CorReg** package uses MCMC to find a set of linear relations in which response variables are never predictors.

***Resizable, scalable, concurrent hash tables
via relativistic programming***
J. Triplett et al.

Hash tables can be resized while in use.

***progenyClust:
an R package for progeny clustering***
C.W. Hu and A.A. Qutub (2016)

To choose the number of clusters (in k -means, etc.), look at their stability, rather than just their compactness or how distinct they are. For instance:

- Cluster the data into C_1, \dots, C_k ;
- Sample N elements from each C_i
- Cluster the resulting data;
- Look at the cooccurrence matrix: let Q_{ab} indicate whether a and b are in the same (new) cluster; we expect this matrix to be approximately block-diagonal;
- Repeat R times, average the Q matrices, and compute

$$\text{Stability} = \frac{P[a \sim' b \mid a \sim b]}{P[a \sim' b \mid a \not\sim b]}$$

where $a \sim b$ (resp. $a \sim' b$) indicates that a and b were (are) in the same old (new) cluster.

Check the R packages: **progenyClust**, **cclust**, **clusterSim**, **Nbclust**, **fpc**, **mclust**.

***mclust 4: clustering,
classification and density estimation
using Gaussian finite mixture models***
L. Scrucca et al. (2016)

The **mclust** package provides EM clustering (**Mclust**) with various variance structures; the best one and the number of clusters can be selected with the BIC (**mclustBIC**) or the *integrated complete likelihood criterion* (ICL) to limit cluster overlap (**mclustICL**); the data can be plotted after dimension reduction (**MclustDR**). The likelihood ratio test (with a bootstrap-estimated p -value) can compare models with k and $k + 1$ clusters. Model-based hierarchical clustering merges the two clusters with the smallest likelihood decrease; it can be used to initialize the EM algorithm./ Mixture models can also estimate densities (**densityMclust**) or classify observations (**mclustDA** – each class is a mixture of one (EDDA) or more Gaussians).

***Conditional fractional Gaussian fields
with the package FieldSim***
A. Brouste et al. (2016)

The **FieldSim** package simulates fractional Gaussian fields on the line, plane or sphere, and their variants (fractional and multifractional Brownian fields, Brownian sheets, anisotropic fields):

$$R(x, y) = \frac{1}{2}(\|x\|^{2H} + \|y\|^{2H} - \|x - y\|^{2H})$$

$$R(x, y) \propto \|x\|^{H(x)+H(y)} + \|y\|^{H(x)+H(y)} - \|x - y\|^{H(x)+H(y)}$$

$$R(x, y) = \frac{1}{2^d} \prod_{i=1}^d (|x|_i^{2H_i} + |y|_i^{2H_i} - \|x_i - y_i\|^{2H_i})$$

$$R(x, y) = v(x) + v(y) - v(x - y).$$

***keyplayer: an R package for locating
key players in social networks***
W. An and Y.H. Liu (2016)

There are many measures of centrality: degree, closeness (harmonic mean of pairwise distances), betweenness (proportion of shortest paths that go through a node i), eigenvector, M -reach degree, M -reach closeness (similar to degree and closeness, with the node of interest replaced by the set of nodes at most M steps away – there are two such neighbourhoods, $\{j : d(i, j) \leq M\}$ and $\{j : d(j, i) \leq M\}$: use both), fragmentation (harmonic mean after removing node i), diffusion ($\sum_{k \leq T} P^k$)**1**.

Centrality measures can be aggregated for a group of nodes: min, max, sum or $1 = \prod_G (1 - c_i)$ if the centralities can be interpreted as probabilities.

***CryptRndTest: an R package for testing
the cryptographic randomness***
H. Demirhan and N. Bitirim (2016)

More tests of randomness for a sequence of bits, letters or numbers:

- Adaptive χ^2 : divide the alphabet into subsets and perform a χ^2 test on them;

- Birthday spacing: number of duplicated values of spacings between ordered birthdays in a year of pre-defined length
- Bookstack (?);
- $\gcd(x_{2n}, x_{xn+1})$ should be iid;
- The number of steps to compute these gcd's should also be iid;
- Interpret the sequence of bits as a random walk and look at the distribution of excursions, heights (in excursions) and expansions (max-min);
- Number of different bit patterns (just the number, not their frequency).

clustering.sc.dp: optimal clustering with sequential constraint by using dynamic programming
T. Szkaliczki (2016)

The k -means optimization problem can be solved exactly, in polynomial time, with dynamic programming, in dimension 1 or in dimension n when only consecutive items can form a cluster: let $D[i, m]$ be the best score when clustering x_1, \dots, x_i into m clusters.

Spatio-temporal interpolation using gstat
B. Gräler et al. (2016)

Spatial and temporal dependence structures can be combined in many ways.

$$\begin{aligned} C(h, u) &= C_s(h)C_t(u) \\ C(h, u) &= kC_s(h)C_t(u) + C_s(h) + C_t(u) \\ C(h, u) &= C(\sqrt{h^2 + (\kappa u)^2}) \\ C(h, u) &= C(\sqrt{h^2 + (\kappa u)^2}) + C_s(h) + C_t(u) \end{aligned}$$

simplexreg: an R package for regression analysis of proportional data using the simplex distribution
P. Zhang et al.

The simplex distribution on $[0, 1]$ is

$$\begin{aligned} p(y) &\propto y^{\alpha-1}(1-y)^{\beta-1} \exp \frac{-d_{\alpha\beta}(y; \mu)}{2\sigma^2} \\ d_{\alpha\beta}(y; \mu) &= \mu^{2\alpha-1}(1-\mu)^{2\beta-1} \frac{y(1-y)}{(y-\mu)^2}. \end{aligned}$$

Special cases include the beta distribution ($\sigma \rightarrow \infty$) and the standard simplex distribution ($\alpha = \beta = \frac{1}{2}$).

PerMallows: an R package for Mallows and generalized Mallows models
E. Iruruzki et al. (2016)

The **Mallows model** is a distribution on \mathfrak{S}_n , with applications in recommender systems, information retrieval, preference elicitation:

$$p(\sigma) \propto \exp -\theta d(\sigma, \sigma_0)$$

where σ_0 is the mode, θ a dispersion parameter, and d a distance, e.g.,

- Kendall's τ (or bubble sort) distance: the number of inversions;
- Cayley's distance: the minimum number of transpositions;
- Hamming's distance: the number of positions that disagree;
- Ulam's distance: the number of items not in the longest common subsequence (LCS).

The *generalized Mallows model* (GMM) has a different dispersion parameter θ for each position i , to give more importance to some of them.

The **Bradley-Terry** model for pairwise comparisons is

$$P[i > j] = \frac{p_i}{p_i + p_j}$$

where p_i is the skill of i (it could depend on covariates).

FuzzyStatProb: an R package for the estimation of fuzzy stationary probabilities from a sequence of observations of an unknown Markov chain
P.J. Villacorta and J.L. Verdegay (2016)

A subset A of X is determined by its indicator function $1_A : X \rightarrow \{0, 1\}$.

A fuzzy subset of X is a function $\mu : X \rightarrow [0, 1]$.

A fuzzy number is a function $\mu : \mathbf{R} \rightarrow [0, 1]$ such that the α -cuts $[\mu \geq \alpha]$ are compact, convex, non-empty (for $\alpha \in [0, 1]$), decreasing,

$$\alpha \leq \beta \implies [\mu \leq \beta] \subset [\mu \geq \alpha]$$

and continuous

$$[\mu \geq \alpha] = \bigcap_{\beta < \alpha} [\mu \geq \beta],$$

The estimated transition probabilities of a Markov chain can be modeled as fuzzy numbers, by stacking their confidence intervals, with a crisp constraint that they add up to 1.

gramEvol: grammatical evolution in R
F. Noorian et al. (2016)

A derivation of a context-free grammar (CFG) can be encoded as a sequence of numbers, each indicating which (left-most) rule to apply (if the number is larger than the number of applicable rules, use the modulo operator). Canonical genetic algorithms (on strings) can manipulate those derivations (add a length limit, and ignore the end of the string, once no non-terminals remain: all (sufficiently long) strings encode valid derivations).

Application of *grammatical evolution* (GE) include symbolic regression, feature generation, regular expression discovery, decision tree construction, ensembling (with hyperparameters), etc.

Most other implementations (libGE, AGE, GEVA, PonyGE, PyNevrGer, GEM, GERET) look unmaintained.

Computation of graphlet orbits for nodes and edges in sparse graphs
T. Hočevár and J. Demšar (2016)

To build a vector embedding of nodes in a graph (“graphical n -grams”), count the graphlets (connected induced subgraphs of size at most k containing the node) and the orbits (position of the node in the graphlet, up to automorphism). The `orca` package computes those graphlet and orbit counts efficiently (start with the smallest graphlets and notice that graphlet inclusions induce linear relations between graphlets and orbit counts). The embedding can be used with the `FNN` package to find nodes with a neighbourhood structure similar to that of a query node.

Copula regression spline sample selection models: the R package `SemiParSampleSel`
W. Wojtyś (2016)

To model data missing not at random, try

$x \sim N(\mu, \sigma)$	value
$y \sim N(\nu, 1)$	missingness score
$x \mathbf{1}_{y>0}$	observed

and use a copula for (x, y) .

Laplace’s rule of succession in information geometry
Y. Ollivier

The *sequential normalized maximum likelihood* estimator is the maximum likelihood estimator if the new value y had already been observed

$$\begin{aligned}\theta^{\text{ML}} &= \underset{\theta}{\text{Argmax}} \sum_i \log p_{\theta}(x_i) \\ \theta^{\text{ML}+y} &= \underset{\theta}{\text{Argmax}} \log p_{\theta}(y) + \sum_i \log p_{\theta}(x_i) \\ p^{\text{SNML}}(y) &\propto p_{\theta^{\text{ML}+y}}(y)\end{aligned}$$

(since $\theta^{\text{ML}+y}$ depends on y , $p_{\theta^{\text{ML}+y}}(y)$ is not a probability and has to be normalized).

For exponential families, $\frac{1}{2}(p^{\text{ML}} + p^{\text{SNML}})$ is close to the Bayesian predictor obtained from the Jeffreys prior; it generalizes the “add-one-half” rule (Laplace’s “add-one” rule corresponds to a uniform prior on a Bernoulli distribution).

On the chi square and higher-order chi distances for approximating f -divergences
F. Nielsen (2013)

The Taylor expansion of a (convex) analytic function f gives an expansion of the f -divergence as a series of χ^2 divergences. There are closed form formulas (no integrals) for the (Pearson, Neyman, Vajda) χ^2 divergences for exponential families.

Barycentric subspace analysis on manifolds
X. Pennec (2016)

There are many generalizations of principal component analysis (PCA) to manifolds:

- Tangent PCA: start with the Fréchet mean and do all the computations in its tangent space (this only works if the distribution is sufficiently centered around the mean);
- Principal geodesic analysis minimizes the least squares distances to totally geodesic subspaces (it is often approximated by tPCA);
- Geodesic PCA: find the geodesic best fitting the data (it does not necessarily go through the mean); add a second geodesic orthogonal to the first, etc.

A *barycentric subspace* is the locus of weighted means of $k - 1$ points. The *Fréchet* (*Karcher*) mean is a global (local) minimizer of $\sum w_i d(x, x_i)^2$; the *exponential* mean is a solution of $\sum w_i \overrightarrow{xx_i} = 0$ where $\overrightarrow{xx_i} = \log_x(x_i)$ is the Riemannian logarithm. A Fréchet mean is a Karcher mean, which is an exponential mean.

Barycentric subspace analysis returns the *flag* (nested subspaces of increasing dimensions) of barycentric subspaces minimizing the unexplained variance.

Morphology, geometry and statistics in nonstandard imaging
E. Chevallier (2015)

The pixels of an image are no longer limited to points in a Euclidian space:

- Colours form a (non-flat) Riemannian space;
- With *diffusion tensor imaging* (DTI), voxels are 3×3 positive definite matrices (representing the distribution of the velocities of water molecules in this voxel).

Mathematical morphology is based on two operations:

Erosion	$\varepsilon(I)_p = \inf_{q \in N(p)} I_q$
Dilatation	$\delta(I)_p = \sup_{q \in N(p)} I_q$
Opening	$\gamma = \delta \varepsilon$
Closing	$\phi = \varepsilon \delta$

For arbitrary pixel values, e.g., elements of $[0, 1]^3$, one can try to impose a total order (not on all possible values, but on the values actually taken):

- With space-filling curves;
- By solving the TSP on the set of pixel values;
- By solving a combinatorial optimization problem looking for a total order avoiding pairs of points close in in colour space by distant in the order, but allowing problematic triplets of values provided they occur in different regions of the space.

For pixel values in an unordered finite set of labels, one can define a dilatation and an erosion for each label i , extending or reducing the area occupied by this label (for the erosion, fill in the blanks with, e.g., the nearest value). For $n > 2$ labels, there are n dilatations and n

erosions. This can be generalized to discrete probability distributions on the set of labels (“fuzzy labels”).

To estimate a density (image histogram) on a Riemannian space (e.g., a colour space, with the Riemannian structure given by MacAdam ellipses, or Luo & Rigg’s chromaticity discrimination ellipses (BFD-P), or Berns’s colour difference tolerance (RIT-DuPont), or spaces of Gaussian laws, with the Fisher or Wasserstein (earth’s mover’s) distance) only short geodesics are needed: a local Euclidean approximation suffices.

Logarithmic time one-against-some
H. Daumé III et al.

For extreme multiclass classification (*i.e.*, a very large number of classes), build a decision tree, optimizing the decrease in entropy of the label distribution (but only for nodes whose recall is better than that of their children, to ensure the counts are reliable), and use it to select $O(\log K)$ candidates; then, use one-against-some (instead of the reference, one-against-all).

Bag of tricks for efficient text classification
A. Joulin et al.

Use a linear classifier with Huffman-coding-based hierarchical softmax, bag-of-words and n -grams, and the hashing trick.

Optimizing non-decomposable performance measures: a tale of two classes
H. Narasimhan et al. (2015)

Performance measures for unbalanced classification problems are non-decomposable, and not amenable to large scale optimization algorithms such as stochastic gradient descent (SGD). Try to approximate the objective with a linear function of TPR and TNR (e.g., if it is a ratio of linear functions: F-mean, Jaccard coefficient) or use a primal-dual approach (for concave functions of TPR and TNR: G-mean, H-mean).

Learning stationary time series using Gaussian processes with nonparametric kernels
F. Tobar et al.

Kernel choice encapsulates smoothness, stationarity and periodicity assumptions on the function. The model

$$\begin{aligned}k &\sim \text{GP}(0, k) \\ x &\sim \text{GP}(0, \sigma^2 \delta) \text{ white noise} \\ f &= h * x\end{aligned}$$

is equivalent to

$$\begin{aligned}h &\sim \text{GP}(0, k) \\ f &\sim \text{GP}(0, k(\cdot) * k(-\cdot))\end{aligned}$$

i.e., it defines a prior on nonparametric models. Use a decaying squared exponential kernel,

$$k(t, s) \propto e^{-\alpha t^2} e^{-\alpha s^2} e^{-\gamma(t-s)^2}.$$

Local and global sparse Gaussian process approximations
E. Snelson and Z. Ghahramani

Sparse Gaussian process approximations replace the training set with a smaller set of “inducing points” (either from the training set, or arbitrary points) and can be global or local (for wiggly functions); the local and global approaches can be combined.

Gaussian process kernels for pattern discovery
A.G. Wilson and R.P. Adams (2013)

Stationary kernels k are determined by their spectral density S : $k(t) = \int S(s) e^{2\pi i s' t} ds$. The omnipresent square exponential kernel has a Gaussian spectral density centered at the origin. Modeling the spectral density as a mixture of (non-centered) Gaussians gives a larger variety of kernels.

Pseudo-marginal Bayesian inference for Gaussian processes
M. Filippone and M. Girolami (2014)

To sample from $p(f, \theta|y)$, where θ are hyperparameters, f latent variables, and y observed data, one can alternatively sample from $p(f|\theta, y)$ and $p(\theta|f, y)$ – but the distribution of the hyperparameters is too concentrated and leads to slow convergence and mixing. Instead, the **pseudo-likelihood** approach samples from $p(\theta|y)$, using $p(\theta|y) \propto p(y|\theta)p(\theta)$ and the unbiased estimator

$$\begin{aligned}p(y|\theta) &\approx \frac{1}{N} \sum \frac{p(y|f_i)p(f_i|\theta)}{q(f_i|y, \theta)} \\ f_i &\sim q(\cdot|y, \theta) \text{ importance sampling.}\end{aligned}$$

Distributed Gaussian processes
A.P. Deisenroth and J.W. Ng (2015)

Train local experts on subsets of the data (e.g., using KD-trees), and combine them, e.g., as a mixture of experts, a product of experts, or a Bayesian committee machine (beware, many of those lead to over- or under-confident forecasts).

Thoughts on massively scalable Gaussian processes
A.G. Wilson et al.

Use inducing points and the structure of the covariance matrix: Toeplitz (regular 1-dimensional grid), Kronecker (arbitrary n -dimensional grid) or block-Toeplitz with Toeplitz blocks.

Mollifying networks

C. Gulcehre et al.

To minimize a badly-behaved function f , **mollify** it, *i.e.*, minimize $f * k_\varepsilon$, where k_ε is \mathcal{C}^∞ , $f * k_\varepsilon$ is convex, and $f * k_\varepsilon \rightarrow f$ when $\varepsilon \rightarrow 0$. If $f * k_\varepsilon$ is difficult to compute, use a Monte Carlo approximation – this is equivalent to adding noise to the input.

In addition, for neural nets:

- For each unit, either use the (noisy) activation function, or let the signal pass through, randomly (cf. ResNet);
- Bound the activation function using its linear approximation (if the noise is high).

The concave-convex procedure (CCCP)

A.L. Yuille and A. Rangarajan (2003)

To minimize the sum of a convex and concave function, $E = E_+ + E_-$, the CCCP build a sequence x_0, x_1, \dots such that $\nabla E_+(x_{n+1}) = -\nabla E_-(x_n)$; it monotonically decreases E . If finding x such that $\nabla E_+(x) = -\nabla E_-(x_0)$ is not obvious, iterate $x_{n+1} \leftarrow \text{Argmin}_x E_+(x) + \langle x, \nabla E_-(x_0) \rangle$ (these are convex problems).

Most functions are of this form: if the Hessian of E is bounded, adding a sufficiently large strictly convex function λF gives a convex function and a decomposition $E = (E + \lambda F) + (-\lambda F)$.

Applying the CCCP to minimize $E_+(x) + E_-(x)$ is equivalent to minimizing $E_+(x) + \langle x, y \rangle + E_-^*(y)$ wrt x and y alternatively, where E_-^* is the Fenchel-Legendre transform (aka convex conjugate) of E_- .

Hyperparameter optimization of deep neural networks

using non-probabilistic RBF surrogate model

I. Ilievski et al.

Non-probabilist alternative to Bayesian optimization using radial basis function (RBF) surrogates.

Hyperbolic geometry of complex networks

D. Krioukov et al. (2010)

The curvature of a graph can be defined by looking at how the number of nodes at distance (at most) d of a given node changes with d , and comparing it with the length (area) of a circle (disk) of radius r in the hyperbolic plane of constant curvature k . For instance, for a b -ary tree, $n_d \sim b^d$ and $\text{area}_d \sim e^{k^2 d}$.

(The actual model is more complicated and considers nested or overlapping disks in the hyperbolic plane.)

Fast generation of complex networks with underlying hyperbolic geometry

M. von Looz et al.

A random hyperbolic graph is obtained by taking vertices at random in the Poincaré disk and joining pairs

whose distance is below R . Naively computing all pairwise distances is quadratic: use a suitable data structure, e.g., a *polar quad-tree*.

Cryptographic secure pseudo-random bits generation: the Blum-Blum-Shub generator

P. Junod (1999)

The BBS cryptographic RNG is defined by

$$p, q \equiv 3(4) \quad \text{large primes}$$

$$n = pq$$

$$s \in \mathbf{Z}/(n), \quad s \neq 0$$

$$x_0 = s^2$$

$$x_n = x_{n-1}^2$$

$$z_n = \text{parity}(x_n)$$

(for such n 's, squaring is a permutation of quadratic residues (*i.e.*, squares) modulo n).

Object spreadsheets: a new computational model for end-user development of data-centric web applications

M. McCutchen et al. (2016)

The spreadsheet computation model (data flow, graph computations) is not adapted to spreadsheets: it considers tabular or hierarchical data as unstructured and has no relational operators to manipulate data.

Changing that computation model to focus on columns rather than cells, and separating the tabular (developer) view from the user interface (UI) could help build data-centric applications (which are often, currently, shared Google spreadsheets).

Bayesian learning via stochastic gradient Langevin dynamics

M. Welling and Y.W. Teh (2011)

Langevin dynamics can be seen as gradient descent with added noise; it “converges” to the whole posterior distribution instead of just its mode. Since stochastic gradient descent is already a noisy gradient descent, it can be modified, by adding the right amount of noise, to sample from the posterior.

A piecewise deterministic scaling limit of lifted Metropolis-Hastings in the Curie-Weiss model

J. Bierkens and G. Roberts (2016)

Given a Markov chain $P(x, y)$ on a state space S , e.g., a Metropolis-Hastings chain, and a function $\eta : S \rightarrow \mathbf{R}$,

$$\begin{pmatrix} T^+ & \text{diag } T^{+-} \\ \text{diag } T^{-+} & T^- \end{pmatrix}$$

$$\begin{aligned}
T^+(x, y) &= P(x, y) \mathbf{1}_{\eta y \geq \eta x} \\
T^-(x, y) &= P(x, y) \mathbf{1}_{\eta y \leq \eta x} \\
T^{+-}(x) &= \left(\sum_{y \neq x} T^+(x, y) - T^-(x, y) \right)_+ \\
T^{-+}(x) &= \left(\sum_{y \neq x} T^-(x, y) - T^+(x, y) \right)_-
\end{aligned}$$

defines a (non-reversible, “lifted”) Markov chain on $S \times \{\pm 1\}$ with the same stationary distribution.

***Irreversible Monte Carlo algorithms
for efficient sampling***
K.S. Turitsyn et al. (2010)

***Piecewise deterministic Markov processes,
applications in biology***
R. Yvinec

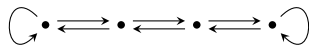
Piecewise deterministic Markov processes (PDMP, an example of a non-diffusion) are ODEs, switching at random (Poisson) times, e.g., a cell growing (ODE) and dividing (Poisson), or a gene, turned on or off (Poisson) regulating some reaction (ODE).

***Scalable MCMC for large data problems using
data subsampling and the difference estimator***
M. Quiroz et al.

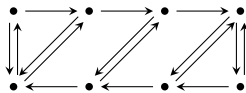
An approximate acceptance probability, computed from a subsample (in a big data problem), is sufficient to ensure a Metropolis-Hastings chain has the desired stationary distribution.

Liftings of tree-structured Markov chains
T.P. Hayes and A. Sinclair

Lifting a markov chain (replacing each state with a set of states) can improve mixing time. For instance, to the diffusive behaviour of



one can add momentum.



***The zig-zag process and super-efficient
sampling for Bayesian analysis of big data***
J. Bierkens et al. (2016)

The *zigzag process* is a continuous-time piecewise deterministic Markov process (PDMP) on $\mathbf{R}^d \times \{\pm 1\}^d$: $\xi : \mathbf{R}_+ \rightarrow \mathbf{R}^d$ is piecewise linear, with $\xi' = \theta$, $\theta : \mathbf{R}_+ \rightarrow \{\pm 1\}^d$; a single component of θ flips at random times, at rates $\lambda_i(\xi_t, \theta_t)$; it can be sampled with Poisson thinning. Adding momentum makes the chain

non-reversible (Hamiltonian Monte Carlo (HMC) also adds momentum, but remains reversible) but aids mixing. The *bouncy particle sample* is similar, but the momentum is arbitrary: $\theta_t \in \mathbf{R}$. If the rates are strictly positive, the chain is ergodic.

Given a target probability density π , the canonical zigzag process is defined by

$$\begin{aligned}
\psi &= -\log \pi + \text{constant} \\
\lambda_i(\xi, \theta) &= [\theta_i \partial_i \psi(\xi)]_+.
\end{aligned}$$


When $\psi = \sum_j \psi_j$ (e.g., in big data applications), *sub-sampling* (using a different ψ_i at each iteration) does not change the invariant distribution (but mixing deteriorates); *control variates*, e.g., writing

$$\psi(\xi) = \psi(\xi^*) + \sum_j \psi_j(\xi) - \psi_j(\xi^*),$$

(as in variance-reduced stochastic gradient) improves mixing.

***Bayesian linear mixed models using Stan:
a tutorial for psychologists, linguists
and cognitive scientists***
T. Sorensen et al. (2016)

***DehazeNet: an end-to-end system
for single image haze removal***
B. Cai et al.

Use a convolutional neural net with convolutions of different sizes and bilateral ReLUs  to estimate the transmission map (the haze intensity – easy if you have a depth map) and de-haze. Commonly-used features include:

- Dark channel, $D(x) = \min_{y \in N_x} \min_{c \in \{r, g, b\}} I_c(y)$, where x is the pixel position and N_x its neighbourhood (in haze-free patches, at least one colour channel is close to zero – except the sky);
- Maximum contrast $C(x) = \max_{y \in N_x} \sum_{z \in N_y} \|I(z) - I(y)\|^2$;
- Colour attenuation $A(x) = I^{\text{value}}(x) - I^{\text{saturation}}(x)$;
- Hue disparity $H(x) = |I_{\text{si}}^{\text{hue}}(x) - I^{\text{hue}}(x)|$ where $I_{\text{si}}^c(x) = \max\{I^c(x), 1 - I^c(x)\}$, $c \in \{r, g, b\}$.

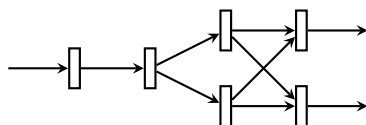
***A hierarchical latent variable
encoder-decoder model for generating dialogues***
I.V. Serban et al.

Add latent (stochastic, unobserved) variables to a RNN to capture larger structures (in a usual RNN, the input and the output are stochastic, but they only control the local structure of the sentences).

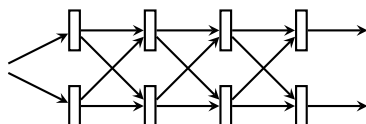
Cross-stitch networks for multi-task learning

I. Misra et al.

When using neural nets for different tasks on the same input, one needs to choose when to split the two networks.



Instead, cross-stitch units use two representations at each level, but blends them.



$$\begin{pmatrix} \tilde{x}_A \\ \tilde{x}_B \end{pmatrix} = \begin{pmatrix} \alpha_{AA} & \alpha_{AB} \\ \alpha_{BA} & \alpha_{BB} \end{pmatrix} \begin{pmatrix} x_A \\ x_B \end{pmatrix}$$

LSTM-based deep learning models for non-factoid answer selection

M. Tan et al. (2016)

Model questions with a CNN on top of a bidirectional RNN with LSTM units and an attention model; use a similar network for potential answers; compare them with cosine similarity.

Bridging nonlinearities and stochastic regularizers with Gaussian error linear units

D. Hendrycks and K. Gimpel

Dropout randomly sets the outs to zero: it is a stochastic zero map.

Zero-out randomly lets the input pass through: it is a stochastic identity map.

ReLU is a deterministic zero or identity map, depending on the sign of the input.

The *stochastic zero-identity* map applies the identity with probability $\Phi(x) = P[Y \leq x]$, $Y \sim N(0, 1)$, and the zero map otherwise; in expectation, it is the (non-convex, non-monotonic) *Gaussian error linear unit* $\text{GELU}(x) = x\Phi(x)$.

Very deep convolutional networks for natural language processing

A. Conneau

On character streams, very deep CNNs outperform (shallow) LSTM RNNs.

Decoupled neural interfaces using synthetic gradients

M. Jaderberg et al.

Instead of waiting for the gradients in back-propagation, try to predict them using local information.

Surprisal-driven feedback in recurrent networks

K. Rocki

Monitoring the discrepancy between forecasts and actual observations can improve RNNs.

Densely connected convolutional networks

G. Huang et al.

Connect each layer to each other layer (there are $\frac{1}{2}n(n-1)$ connections instead of n).



LIFT: learned invariant feature transform

K.M. Yo et al. (2016)

Three CNNs, in a sequential pipeline, image \rightarrow detector \rightarrow crop \rightarrow orientation estimator \rightarrow rotate \rightarrow descriptor, trained on SIFT features.

XNOR-Net: ImageNet classification using binary convolutional neural nets

M. Rastegari et al. (2016)

Binary networks (binary weights, or both binary weights and inputs) are much faster (thanks to bitwise operations) and sufficiently accurate.

Synthesizing the preferred inputs for neurons in neural networks via deep generator networks

A. Nguyen et al. (2016)

Activation maximization (finding the image that excites a neuron the most) yields uninterpretable results unless one uses a *natural image prior*, e.g., from a GAN.

Virtual worlds as proxy for multi-object tracking analysis

A. Gaidon et al.

Virtual worlds provide photo-realistic labeled data.

DeMo Dashboards: visualizing and understanding genomic sequences using deep neural networks

J. Lanchantin et al.

Interpreting deep neural nets is more important in biomedical application, e.g., predicting transcription factor binding sites (TFBS) using a CNN on top of a bidirectional RNN with LSTM units:

- Saliency: importance of the nucleotide, *i.e.*, derivative of the output wrt the input;
- Score, as a time series;
- Sequence that maximizes the TFBS probability.

***Faster training of very deep networks
via p -norm gates***
T. Pham et al. (2016)

P-norm gates,

$$y = \alpha_1 x_1 + \alpha_2 x_2, \quad \alpha_1^p + \alpha_2^p = 1, \quad p \geq 1,$$

are between residual gates ($p = \infty$, $\alpha_1 = \alpha_2 = 1$) and highway gates ($p = 1$).

Layer normalization
J.L. Ba et al.

Batch normalization normalizes the output of a single neuron within a mini-batch; *layer normalization* normalizes the outputs of all neurons in a layer, for a single training case; it is applicable to RNNs.

***RAISR: rapid and accurate
image super resolution***
Y. Romano et al.

Image upscaling (“super-resolution”) algorithms used to be simple interpolators (nearest neighbour, bilinear, bicubic), but more recent algorithms use an *image prior* and learn a non-linear map

low-resolution patch \mapsto high-resolution patch

e.g., using sparse coding or CNNs. To speed up the computations, one can instead learn a (linear) map

bilinear upscaled LR patch \mapsto (sharpened) HR patch

refining a cheaply-upscaled image. To make the algorithm more adaptive, cluster the image patches (coarsely, by hashing the local gradient statistics, angle, strength, coherence, from an eigen analysis) and learn a filter for each cluster.

***A+: adjusted anchored neighborhood
regression for fast super-resolution***
R. Timofte et al.

Sparse coding for image super-resolution.

***Image super-resolution
using deep convolutional networks***
C. Dong et al.

CNNs for image super-resolution.

***Enriching word vectors
with subword information***
P. Bojanowski et al.

To compute word embeddings of rare words or in morphologically rich languages, represent words as bags of n -grams.

***Charagram: embedding words and sentences
via character n -grams***
J. Wieting et al.

Character n -gram count vector embedding for sentences, trained with the paraphrase pairs database (for word or sentence similarity), or the Penn treebank (for POS tagging).

Correlated topic models
B.M. Blei and J.D. Lafferty

Correlation between topics can be introduced in latent Dirichlet analysis (LDA) by replacing the Dirichlet distribution

$$Y_i \stackrel{\text{iid}}{\sim} \Gamma(\alpha_i, \theta) \\ X = Y/(Y'\mathbf{1}) \sim \text{Dir}(\alpha_1, \dots, \alpha_n)$$

with a **logistic normal distribution**

$$Y \sim N(\mu, \Sigma) \\ X = Y/(Y'\mathbf{1}).$$

***Content-based recommendations
with Poisson factorization***
P. Gopalan et al.

The collaborative topic **Poisson factorization** is suited to sparse, long-tailed data, and amenable to variational inference.

k : topics

v : words

d : documents

u : users

$$\beta_{vk} \sim \Gamma$$

$$\theta_{dk} \sim \Gamma$$

$$w_{dv} \sim \text{Poisson}(\theta'_d \beta_v)$$

$$\eta_{uk} \sim \Gamma$$

$$\varepsilon_{dk} \sim \Gamma$$

$$r_{ud} \sim \text{Poisson}(\eta'_u(\theta_d + \varepsilon_d))$$

Learning to search better than your teacher
K.W. Chang et al. (2015)

Structured prediction problems (e.g., building the parse tree of a sentence) can be turned into reinforcement learning problems by expressing the prediction as a sequence of actions (e.g., growing the tree). **Learning to search** tries to replicate a reference policy (*i.e.*, a map state features \rightarrow action) instead of solving the full reinforcement learning problem. Locally optimal learning to search (LOLS) tries to modify the reference policy (no longer assumed optimal) to improve the performance.

**Deep unsupervised learning
using nonequilibrium thermodynamics**
J. Sohl-Dickstein et al. (2015)

To model a probability distribution (e.g., pixels in the MNIST dataset), find a diffusion that transforms it into a tractable one (e.g., Gaussian).

**Stochastic differential equation
mixed effects models for tumor growth
and response to treatment**
J.L. Forman and U. Picchini

Biological phenomena are often modeled by ODEs, but SDEs may be a better choice; as state space models, they can be estimated with particle filters (sequential Monte Carlo, SMC). Mixed effects SDE (SDEM) can be estimated with approximate Bayesian computations (ABC) or *synthetic likelihood* (similar to ABC, but we assume the summary statistics are Gaussian, *i.e.*, we only look at the mean and variance: it is a simulated method of moments).

**A note on the validity of cross-validation
for evaluating time series prediction**
C. Bergmeir et al. (2015)

Even in presence of auto-correlation, most estimators remain consistent (the variance of the limiting distributions and the resulting tests are affected, though); likewise, cross-validation can be applied to time series without any modification.

**Faster principal component regression via
optimal polynomial approximation to $\text{sgn}(x)$**
Z. Allen-Whi and Y. Li (2016)

To project a vector $v \in \mathbf{R}^n$ to the subspace spanned by the eigenvectors of A with eigenvalues at least λ_0 , without computing the eigenvectors, compute $g(A) = \text{sign}(\lambda - \lambda_0 - \varepsilon)$; use a polynomial approximation of g .

**DART: dropouts meet
multiple additive regression trees**
K.V. Rashmi and R. Gilad-Bachrach (2015)

Boosted regression trees, with dropout of complete trees.

Tensors and their eigenvectors
B. Sturmfels (2016)

A symmetric $n \times n \times \dots \times n$ tensor $T = (t_{i_1 \dots i_d})$ can be seen as a homogeneous polynomial of degree d in n variables $T = \sum t_{i_1 \dots i_d} x_{i_1} \dots x_{i_d}$ (symmetric matrices correspond to bilinear forms) and, therefore, as a hypersurface in \mathbf{P}^{d-1} .

A vector $v \in \mathbf{R}^n$ is an eigenvector of T if $(\nabla T)v = \lambda v$ for some λ . Algebraic geometry can help count the number of eigenvalues or singular values of (symmetric) tensors.

Code-based cryptography
I. Márquez-Corbella (FUN 2016)

Quantum computers can solve the integer factorization and the discrete log problems in reasonable time. Code-based cryptography relies on NP-completeness and could form the base of **post-quantum cryptosystems**. A linear code is a subspace of \mathbf{F}_q^n of dimension k , defined either as the kernel of a *parity check matrix* H or as the image of a linear map $G : \mathbf{F}_q^k \rightarrow \mathbf{F}_q^n$ of rank k . The *syndrome* of a received word x is Hx ; it is also the syndrome of the error. In the *McEliece cryptosystem*, the public key is a generator matrix G with added noise, and the private key is an efficient decoding algorithm for G . In the *Niederreiter cryptosystem*, the public key is a parity check matrix and the private key is an efficient syndrome decoder.

While promising, most code-based cryptosystems have been broken – Goppa codes are the main exception.

Determinantal point processes
A. Kulesza and B. Taskar (2012)

Determinantal point processes (DPP) are distributions over subsets $Y \subset \mathcal{Y}$ of the form

$$\begin{aligned} P(A \subset Y) &= \det(K_A) & 0 \preceq K \preceq I \\ P(Y = Y) &\propto \det(L_Y) & L \succcurlyeq 0 \\ P(Y = Y) &= \frac{\det(L_Y)}{\det(L + I)}. \end{aligned}$$

The two formulations are related by $K = I - (L + I)^{-1}$ and the first one is slightly more general (it allows $P(Y = \emptyset) \neq 0$).

DPPs favour diverse sets:

$$\begin{aligned} P(i, j \in Y) &= K_{ii}K_{jj} - K_{ij}^2 \\ &= P(i \in Y)P(j \in Y) - K_{ij}^2 \\ &< P(i \in Y)P(j \in Y) \end{aligned}$$

Applications of DPPs include ranking search results or summarizing documents by extracting a few sentences. They remain tractable until $|\mathcal{Y}| = 10,000$.

To sample from a DPP:

- Compute the eigen decomposition $L = \sum \lambda_n v_n v_n'$;
- Select a subset V of eigenvectors, including v_i with probability $\lambda_i/(\lambda_1 + 1)$;
- Select $i \in \mathcal{Y}$ with probability

$$\frac{1}{|V|} \sum_{v \in V} (v' e_i)^2;$$

- Replace V with an orthonormal basis of $e_i^\perp \cap \text{Span } V$;
- Repeat until V is empty.

Finding the model of a DPP is NP-hard. The function $\log P$ is submodular (easy to minimize, hard to maximize – the greedy algorithm for approximate maximization only works for *monotone* submodular functions).

Alternatives to DPP include:

- Matérn repulsive process: start with a Poisson point process and remove points more than ε apart (either both points, or just the last one)
- Pairwise Markov process

$$P(\mathbf{Y} = Y) \propto \prod_{i \in Y} \psi_1(i) \prod_{i,j \in Y} \psi_{12}(i,j);$$

- Area interaction process

$$P(\mathbf{Y} = Y) \propto \gamma^{-\sum_{y \in Y} \text{Area } B(y, \varepsilon)};$$

- Determinants can be replaced with permanents, immanents, α -determinants or hyperdeterminants;
- Quasi-random (low discrepancy) processes.

The L matrix can be written $L = B'B$, $B = U \text{diag}(q)$, where q measures the (unary) quality of the items, and the columns ϕ_i of U are (normalized) features for item i .

DPPs are similar to Markov random fields (MRF), but they allow negative interactions – inference for pairwise MRFs is tractable, approximately, provided the potentials are associative, *i.e.*, endpoints tend to have the same value.

For sampling, normalization, marginalization, the dual representation $C = BB'$, instead of the larger $L = B'B$, is often sufficient (you still need B , though).

The likelihood can be expressed as a single determinant

$$\begin{aligned} P(Y) &= \frac{\det L_Y}{\det(L + I)} \\ &= \det(I_Y K + I_{\bar{Y}}(I - K)) \\ &= |\det(K - I_{\bar{Y}})|. \end{aligned}$$

To fit a DPP, use feature vectors ϕ for the items $L_{ij} = q_i q_j \phi_i' \phi_j$, and parametrize the quality $q_i = \exp \frac{1}{2} \theta' \phi_i$ (one can use different features for the quality). Mixtures of DPPs are also easy to fit.

A k -DPP is a DPP constrained with $|\mathbf{Y}| = k$ (it is no longer a DPP): we still have $P(\mathbf{Y} = Y) \propto \det L_Y$, but the normalization constant is different

$$P(\mathbf{Y} = Y) = \frac{\det L_Y}{\sum_{|X|=k} \det L_X} = \frac{\det L_Y}{e_k(\lambda_1, \dots, \lambda_n)}$$

where $\lambda_1, \dots, \lambda_n$ are the eigenvalues of L and e_k is the elementary symmetric polynomial

$$e_k(\lambda_1, \dots, \lambda_n) = \sum_{\substack{J \subset [1, n] \\ |J|=k}} \prod_{i \in J} \lambda_i,$$

which can be computed recursively as $e_k^n = e_k^{n-1} + \lambda_n e_{k-1}^{n-1}$. The sampling algorithm for DPPs can be adapted to k -DPPs (use exactly k eigenvectors).

Structured DPPs (SDPP), *i.e.*, DPPs whose quality and diversity feature factor multiplicatively and addi-

tively,

$$\begin{aligned} q(y) &= \prod_{\alpha \in F} q_\alpha(y_\alpha) \\ \phi(y) &= \sum_{\alpha \in F} \phi_\alpha(y_\alpha) \end{aligned}$$

can be handled with message passing algorithms. They can model paths, grammar derivations, etc.

Diversity networks

Z. Mariet and S. Sra (2016)

To reduce the number of neurons in a neural net, sample a diverse set of neurons using a determinantal point process (DPP), $P(Y) \propto \det L_Y$, with a Gaussian RBF kernel $L_{ij} = \varepsilon + \exp -\beta \|v_i - v_j\|^2$ and fuse the remaining neurons.

Discriminants, resultants and multidimensional determinants

I.M. Gelfand et al. (1994)

Let A be a set of monomials. The *discriminant* of $f \in \text{Span } A$ is an irreducible polynomial in the coefficients of f that vanishes whenever f has a multiple root (x_1, \dots, x_k) with $\forall i \ x_i \neq 0$.

The *resultant* of polynomials f_0, \dots, f_k in k variables is an irreducible polynomial in their coefficients, vanishing whenever they have a common root; it is the discriminant of $f_0(\mathbf{x}) + \sum_{i \geq 1} y_i f_i(\mathbf{x})$.

The *determinant* of a matrix $(a_{ij})_{ij}$ is the discriminant of the bilinear form $\sum a_{ij} x_i x_j$. The *hyperdeterminant* of a tensor $(a_{i_1 \dots i_r})_{i_1 \dots i_r}$ is the discriminant of the homogeneous polynomial $\sum a_{i_1 \dots i_r} x_{i_1} \dots x_{i_r}$.

Learning with submodular functions: a convex optimization perspective

F. Bach (2013)

1. Submodular functions are a combinatorial analogue of *concave* functions, in an attempt to build a theory of “combinatorial convex optimization”. Surprisingly, they also have many *convex* aspects. They generalize matroidal ranks. It turns out that both maximizing (NP-hard) and minimizing (polynomial) submodular functions are interesting problems.

There are three main uses of submodular functions in optimization:

- As the objective of a combinatorial optimization problem;
- To penalize the support of a vector: lasso, group lasso, etc. (structured sparsity);
- To penalize the level set of a vector (clustering).

2. A set function $f : 2^V \rightarrow \mathbf{R}$ is **submodular** if

$$\forall A, B \quad F(A) + F(B) \geq F(A \cup B) + F(A \cap B);$$

it suffices to check (diminishing returns)

$$F(A \cup \{k\}) - F(A) \geq F(A \cup \{k, \ell\}) - F(A \cup \{\ell\}).$$

The following operations preserve submodularity: restriction, extension ($F(A) = G(B \cap A)$), contraction ($F(A) = G(A \cup B) - G(B)$, on $V \setminus B$), partial minimization.

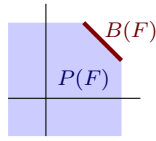
Examples include $|A|$, $\mathbf{1}_{A \neq \emptyset}$, $\#\{i : A \cap G_i \neq \emptyset\}$ where $V = \coprod G_i$ is a partition, or the number of edges from A to $V \setminus A$ in an undirected graph.

A vector $s \in \mathbf{R}^p$ defines a **modular** function

$$s(A) = \sum_{i \in A} s_i, \quad A \subset [1, n].$$

The submodular and **base polyhedron** are

$$\begin{aligned} P(F) &= \{s \in \mathbf{R}^p : \forall A \subset [1, p] \quad s(A) \leq F(A)\} \\ B(F) &= \{s \in P(F) : s(V) = F(V)\} \end{aligned}$$



A **polymatroidal rank function** is a non-decreasing submodular function. A submodular function F can be expressed as a sum of a modular and a polymatroidal function (set $s_k = F(V) - F(V \setminus \{k\})$): $G = F - s$ is polymatroidal).

3. A submodular function F is defined on the vertices of the hypercube $[0, 1]^p$. Its **Lovász extension** (or *Choquet integral*) is the linear interpolation on the $n!$ simplexes corresponding to the orderings of $(x_1, \dots, x_p) \in [0, 1]^p$.

A set function F is submodular iff its Lovász extension is convex; they have the same minima.

The Lovász extension has a probabilistic interpretation:

$$f(w) = E_{u \sim U(0,1)} [F\{i : w_i \geq u_i\}]$$

The *multilinear extension* is neither convex nor concave:

$$\begin{aligned} \tilde{f}(w) &= E_{B \sim w} [F(B)] \\ &= \sum_{A \subset V} F(A) \prod_{i \in A} w_i \prod_{i \notin A} (1 - w_i). \end{aligned}$$

4. A **greedy algorithm** can maximize linear functions on the base polyhedron; in particular, this gives its extreme points.

5. The **convex closure** of $F : 2^V \rightarrow \mathbf{R}$ is the largest convex function $f : \mathbf{R}^p \rightarrow \mathbf{R} \cup \{\infty\}$ such that $\forall A \subset V \quad f(\mathbf{1}_A) \leq F(A)$. It can be obtained by Fenchel biconjugation; it is an extension of F , i.e., $f(\mathbf{1}_A) = F(A)$; it has the same minima. For submodular functions, it is the Lovász extension.

A penalty Ω can add **structural sparsity** to machine learning models,

$$\underset{w}{\text{Argmin}} \text{Loss}(y, x, w) + \lambda \Omega(w).$$

For instance, $\Omega = \#\text{supp } w = \|w\|_0$ or $\Omega(w) = F(\text{supp } w)$, for some set function F , e.g., a partition count.

The convex envelope of $w \mapsto F(\text{supp } w)$, on the ℓ^∞ ball, for a non-decreasing submodular function F , is $\Omega_\infty : w \mapsto f(\|w\|)$, where f is the Lovász extension of F . For instance, the counting function gives the lasso, and the partition count $\sum \|\cdot\|_\infty$, which is not exactly the group lasso – the ball has spurious edges.

Those spurious edges can be removed by replacing $F(\text{supp } w)$ with

$$\frac{1}{p} \|w\|_p^p + \frac{1}{q} F(\text{supp } w), \quad \frac{1}{p} + \frac{1}{q} = 1, \quad p \in (1, \infty).$$

For $\Omega(w) = \#\text{supp } w$, this gives $\Omega_p = \|\cdot\|_p$; for partition counting, this gives the group lasso $\Omega_p = \sum \|\cdot\|_p$.

But those spurious edges can be used: they create clustering of the components of w . For instance,

$$\begin{aligned} F(A) &= \mathbf{1}_{A \neq \emptyset, V}, \\ f(w) &= \text{Max} |w_i - w_j| \end{aligned}$$

creates clusters for the largest and smallest values. Similarly,

$$\begin{aligned} F(A) &= \text{Max}\{|A|, |V \setminus A|\}, \\ f(w) &= \sum_k \text{Min}(n - k, k) |w_{(k+1)} - w_{(k)}| \end{aligned}$$

creates a cluster in the middle and therefore highlights outliers.

6a. Concave functions of cardinality, $F(A) = \phi(|A|)$ are submodular.

6b. *Cut functions*,

$$F(A) = \sum_{\substack{i \in A \\ j \notin A}} d_{ij}$$

(for undirected graphs, they can also be written $F(A) = \frac{1}{2} \mathbf{1}'_A Q \mathbf{1}_A$, where Q is the graph Laplacian) are submodular, with Lovász extension $f(w) = \sum d_{ij} (w_i - w_j)_+$. Examples include the total variation (in one dimension, for a chain $\bullet \cdots \bullet$, or in two dimensions, for a grid \boxtimes) or the isotonic penalty (for the graph of an order relation).

If $G : 2^{V \cup W} \rightarrow \mathbf{R}$ is submodular, then so is its *partial minimization*

$$F : \begin{cases} 2^V & \rightarrow \mathbf{R} \\ A & \mapsto \underset{B \subset W}{\text{Min}} G(A \cup B) \end{cases}$$

A function is *regular* if it is the partial minimization of a cut function.

6c. Given a non-negative function $D : 2^V \rightarrow \mathbf{R}_+$, the *set cover function*

$$F : \begin{cases} 2^V & \rightarrow \mathbf{R} \\ A & \mapsto F(A) = \sum_{B \cap A \neq \emptyset} D(B) \end{cases}$$

is modular and its Lovász extension is

$$f(w) = \sum_B D(B) \max_{k \in B} w_k.$$

The corresponding Ω_p norm is an overlapping group lasso penalty, with no double counting (contrary to $\sum D(B) \|w_B\|_p$). Möbius inversion

$$\begin{aligned} F(A) &= \sum_{B \subset A} G(B) \\ G(A) &= \sum_{B \subset A} (-1)^{|A \setminus B|} F(B) \end{aligned}$$

can recover D ,

$$\begin{aligned} F(V) - F(V \setminus A) &= \sum_{B \subset A} D(B) \\ D(A) &= \sum_{B \subset A} (-1)^{|A| - |B|} (F(V) - F(V \setminus B)). \end{aligned}$$

Set covers can also be defined as $F(A) = \mu(\bigcup_{k \in A} S_k)$, where $S_k \subset W$, for a measure space (W, μ) . Examples include:

- Contiguous subsets of a chain;
- Contiguous 2-element subsets of a chain, to avoid isolated points;
- Convex subsets of a 2-dimension grids, to prefer convex supports;
- Tree structures, to ensure that if a variable is there, then so are its ancestors (gene networks, wavelet bases, models with interactions).

6d. Given a weighted directed graph W , consider flow functions, with sources $S \subset W$ and sinks $V \subset W$. The *maximum net flow* out of $A \subset V$,

$$F(A) = \max_{\phi \text{ flow}} \phi(W, A) - \phi(A, W),$$

is submodular (from the maxflow-mincut theorem, it is the partial minimization of a cut function).

6e. Given discrete random variables (X_1, \dots, X_p) , the *joint entropy* $F(A)$ of $(X_k)_{k \in A}$ defines a modular function. If (X_1, \dots, X_n) is Gaussian, the joint entropy is $H(X_A) = \frac{1}{2} \log \det(2\pi e Q_{AA})$ and $F(A) = \log \det Q_{AA}$ defines a submodular function.

To cluster points x_1, \dots, x_n , compute their (Gaussian, etc.) kernel $k_{ij} = k(x_i, x_j)$ and try to approximate it as two independent ones, *i.e.*,

$$K \approx \begin{pmatrix} K_A & 0 \\ 0 & K_{V \setminus A} \end{pmatrix}$$

for some $A \subset V = [1, n]$. With a prior $p(A) \propto \prod_{k \in A} \eta_k \prod_{k \notin A} (1 - \eta_k)$ (*i.e.*, each point has probability η_k of being in A , and the point memberships are independent), the negative likelihood becomes

$$\begin{aligned} \ell(A) &= D_{\text{KL}} \left(K \left\| \begin{pmatrix} K_A & 0 \\ 0 & K_{V \setminus A} \end{pmatrix} \right. \right) \\ &= - \sum_{k \in A} \log \eta_k - \sum_{k \notin A} \log(1 - \eta_k). \end{aligned}$$

The first term, the mutual information, is submodular, the second, modular. (*Determinantal point processes*, used to ensure diversity in recommender system results, rely on the same idea.)

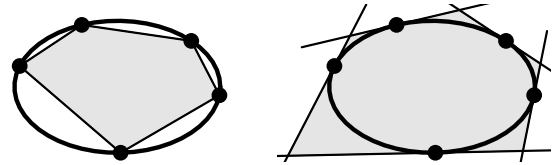
6f. Given a positive semidefinite matrix Q and a function $h : \mathbf{R}_+ \rightarrow \mathbf{R}$, $F(A) = \text{tr } h(Q_A A)$ defines a submodular function for some choices of h (concavity is not enough), e.g., $h(\lambda) = \lambda^p$, $p \in [0, 1]$.

7. To minimize a convex Lipschitz function on a convex set:

- Projected gradient;
- *Ellipsoid*: if you cut an ellipsoid in half, it can be circumscribed by a smaller one (slow in practice);
- *Kelley's method* is a **bundle method**: it keeps the information from all the previous iterations. The points previously examines give a piecewise lower bound of the objective function: Kelley's method minimizes this lower bound;
- The ellipsoid method creates a lot of empty space: we can consider polytopes instead, and their centers of gravity; the center of gravity is not efficiently computable, but can be replaced with the *analytic center*. The analytic center of $\{w : \forall i \ a'_i w \leq b_i\}$, $\text{Argmin}_w \sum -\log(b_i - a'_i w)$, can be computed with Newton's algorithm.

To minimize $\Psi + h$, where Ψ is smooth and strongly convex, and h convex (non-smooth) and positively homogeneous, notice that h can be written $h(w) = \sup_{s \in K} s'w$, for some convex set K .

- *Simplicial methods* are bundle methods that replace h with a piecewise approximation, inner (convex hull) or outer,



and iterates

$$w_{t+1} \leftarrow \underset{w}{\text{Argmin}} \Psi(w) + \max_{1 \leq i \leq t} s'_i w, \quad s_i = h'(w_i).$$

- *Proximal methods* (aka proximal gradient, forward-backward splitting, iterative shrinkage thresholding) replace Ψ with a first-order approximation and add a further penalty to avoid moving too fast

$$\underset{w}{\text{Min}} \Psi(w_0) + (w - w_0)' \Psi'(w_0) + h(w) + \lambda \|w - w_0\|_2^2$$

(there is one optimization at each step (the proximal operator) but, for many penalties, it can be computed in closed form)

Quadratic programs can be solved with an **active set** method, based on the following remarks: if the indices of the non-zero components (“active set”) are known, the problem is easy to solve; it is easy to check if a solution is optimal and, if not, to find a variable to add

to or remove from the active set. This applies, for instance, to least squares with a polyhedral penalty (the epigraph of h is a polyhedron – minimizing h amounts to solving a linear program).

8. If f is the Lovász extension of a submodular function F , the *separable optimization* problem

$$\text{Minimize}_{w \in \mathbf{R}^p} f(w) + \sum \psi_i(w_i)$$

(proximal problem) has dual

$$\text{Maximize}_{s \in B(F)} - \sum \psi_i^*(-s_i).$$

It can be solved by solving the family of submodular optimization problems

$$A^\alpha = \text{Argmin}_{A \subset V} F(A) + \sum_{i \in A} \psi'_i(\alpha)$$

(the second term is the modular function $\psi'(\alpha) \in \mathbf{R}^p$) and setting

$$u_i = \sup\{\alpha \in \mathbf{R} : i \in A^\alpha\}.$$

Since $\alpha < \beta \implies A^\beta \subset A^\alpha$, solving the initial convex optimization problem gives the solution to all those submodular optimization problems (the solution is not unique: $\{i : u_i > \alpha\}$ and $\{i : u_i \geq \alpha\}$ are the minimal and maximal minimizers).

9. The *divide-and-conquer* algorithm solves the dual problem

$$t \leftarrow \text{Argmin}_{t: t(V)=F(V)} \sum_i \psi_i^*(-t_i)$$

$$A \leftarrow \text{Argmin} F - t$$

If $F(A) = t(A)$, return t

$$s_A \leftarrow \text{Argmin}_{s \in B(F_A)} \sum_{i \in A} \psi_i^*(-s_i)$$

$$s_{V \setminus A} \leftarrow \text{Argmin}_{s \in B(F^A)} \sum_{i \notin A} \psi_i^*(-s_i)$$

return $(s_A, s_{V \setminus A})$

where $F_A = F|_A$ and $F^A(B) = F(A \cup B) - F(A)$ (contraction).

For quadratic separable problems, *i.e.*, $\psi_i(w_i) = \frac{1}{2}w_i^2$, the dual is $\text{Min}_{s \in B(F)} \frac{1}{2} \|s\|_2^2$. The **Frank-Wolfe minimum norm point** algorithm solves the optimization problem

$$\begin{aligned} &\text{Find} && \eta \\ &\text{To minimize} && \|\sum \eta_i x_i\|_2^2 \\ &\text{Such that} && \eta \geq 0, \eta' \mathbf{1} = 1 \end{aligned}$$

(*i.e.*, the point in the convex hull of the x_i 's closest to the origin) where the x_i 's are the extreme points of $B(F)$ (they do not have to be explicitly computed), using an active set algorithm.

To minimize a convex function on a convex polytope, the **conditional gradient** approximates it with a linear function, finds the minimizer, and moves towards it.

10. Combinatorial algorithms to minimize a submodular function F use

$$\text{Min}_{A \subset V} F(A) = \text{Max}_{s \in B(F)} s_-(V) = F(V) - \text{Min}_{s \in B(F)} \|s\|_1,$$

where $(s_-)_k = \text{Min}\{s, 0\}$: for instance, the simplex can minimize s_- on $B(F)$.

Instead, one can use convex optimization algorithms to solve $\text{Min}_{w \in [0,1]^p} f(w)$:

- Ellipsoid method;
- Subgradient

$$s_t \leftarrow \text{Argmax}_{s \in B(F)} w'_t s$$

$$w_{t+1} \leftarrow \text{Project}_{[0,1]^p}(w_t - \gamma_t s_t);$$

- Conditional gradient;
- Analytic center cutting plane (ACCP – in practice, this performs best).

One could also solve $\text{Min}_{s \in B(F)} \frac{1}{2} \|s\|_2^2$ or, equivalently, $\text{Min}_{w \in [0,1]^p} f(w) + \frac{1}{2} \|w\|_2^2$ (minimum norm point).

11. Maximizing submodular functions is NP-hard, but, for non-decreasing ones, the **greedy algorithm** (start with $A = \emptyset$ and iteratively add $k \in V \setminus A$ that maximizes $F(A \cup \{k\})$) has a performance guarantee: $F(A) \geq (1 - e^{-1})F(A^*)$. In general, local search (look at $A \cup \{k\}$ and $A \setminus \{k\}$) can help.

Submodular maximization can be expressed as a (non-convex) continuous optimization

$$\text{Max}_{A \subset V} F(A) = \frac{1}{2} F(V) + \frac{1}{2} \text{Max}_{s \in B(F)} \|s\|_1.$$

Differences of submodular functions $F - G$ can be minimized (locally) by iterating

- Find $s \in \mathbf{R}^p$ (*i.e.*, a modular function) such that $s(A) = G(A)$ and $\forall B \ s(B) \leq G(B)$;
- Update $A \leftarrow \text{Argmin}(F - s)$.

The minimum has a geometric interpretation, in terms of Hausdorff distance

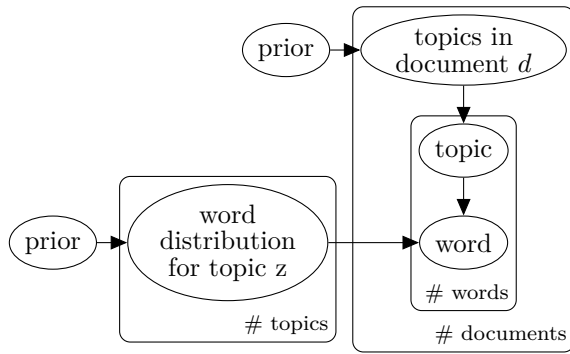
$$\begin{aligned} \text{Min}_{A \subset V} F(A) - G(A) = \\ \frac{1}{2} (F(V) - G(V)) - \frac{1}{2} \text{Min}_{s \in B(G)} \text{Max}_{t \in B(F)} \|t - s\|_1. \end{aligned}$$

Probabilistic topic models M. Steyvers and T. Griffiths

Topic models can be expressed as matrix factorizations:

$$\begin{matrix} \text{cooccurrences} & = & \text{mixture components} & \times & \text{mixture weights} \\ \text{word} \times \text{document} & & \text{word} \times \text{topic} & & \text{topic} \times \text{document} \end{matrix}$$

but, contrary to LSA (SVD), there are constraints on the matrices (probabilities are non-negative and sum to 1) and there is a prior. They can be estimated via Gibbs sampling.



Rethinking LDA: why priors matter H.M. Wallach et al.

Prefer asymmetric Dirichlet priors for the document-topic distribution.

Dynamic topic models D.M. Blei and J.D. Lafferty (2006)

Time-dependent topic models, in which the logarithms of the parameters of the Dirichlet distributions follow random walks, can be estimated via variational inference. (The topics are constant, but their vocabulary changes: for instance, in the “atomic physics” topic, “matter” declined and “quantum” rose in the 20th century.)

From word embeddings to document distances M. J. Kusnet et al. (2015)

To compute the distance between two sentences or texts, (e.g., to retrieve the document closest to a given one), replace them with clouds of points using a vector embedding and use the *earth moving distance* (EMD) between the two clouds. Approximations of the EMD include:

- The distance between the centroids (to quickly identify potential candidates);
- A relaxed (greedy) EMD, in which the whole mass of each word is moved towards the closest word.

Trans-gram, fast cross-lingual word embeddings J. Coulmance et al. (2016)

The skipgram model can also deal with multilingual aligned sentences by defining the context of a word as the set of words in the aligned sentences.

Partially collapsed Gibbs samplers: theory and methods D.A. van Dyk and T. Park (2008)

In some of the steps of a Gibbs sampler, integrating out (marginalizing) some of the variables instead of conditioning on them can improve convergence. Care should be taken to preserve the stationary distribution:

- Move some of the variables from being conditioned to being sampled;

- Re-order the steps so that those new variables are not used;
- Marginalize them.

Example:

$$\begin{array}{cccc} W|XYZ & W|XYZ & WY|XZ & Y|XZ \\ X|WXY & X|WXY & WZ|XY & Z|XY \\ Y|WXZ & WY|XZ & W|XYZ & W|XYZ \\ Z|WXY & WZ|XY & X|WXY & X|WXY \end{array} \rightsquigarrow$$

Metropolis-Hastings within partially collapsed Gibbs samplers D.A. van Dyk and X. Jiao (2014)

Naive use of a Metropolis Hastings step in a PCG sampler (when no closed form distribution is available) often changes the stationary distribution.

A field guide to forward-backward splitting with a Fasta implementation T. Goldstein et al. (2014)

To minimize $f + g$, where f is convex and smooth and g convex, the **proximal gradient** (forward-backward splitting, FBS) method is

$$\begin{aligned} \hat{x}_{k+1} &\leftarrow x_k - \tau \nabla f(x_k) \\ x_{k+1} &\leftarrow \text{prox}_g(\hat{x}_{k+1}, \tau) \end{aligned}$$

where

$$\text{prox}_g(x, \tau) = \underset{z}{\text{Argmin}} \tau g(z) + \frac{1}{2} \|x - z\|^2.$$

If g is differentiable, the second step is a “backward gradient step”, $x_{k+1} \leftarrow \hat{x}_{k+1} - \tau \nabla g(x_{k+1})$: the gradient is evaluated at the new point.

Applications include

- Constrained optimization $\text{Argmin}_{x \in C} f(x)$; the proximal operator of the indicator function of C is the projection, and the algorithm is the projected gradient;
- Lasso: $\text{Argmin}_{\|x\|_1 \leq \lambda} \frac{1}{2} \|Ax - b\|^2$;
- Penalized regression

$$\text{Argmin}_x \frac{1}{2} \|Ax - b\|^2 + \mu \|x\|_1;$$

the proximal operator is soft-thresholding,

$$\text{prox}_{\|\cdot\|_1}(x, \tau) = \text{sign}(x)(|x| - \tau)_+;$$

- Democratic representation

$$\text{Argmin}_x \frac{1}{2} \|Ax - b\|^2 + \mu \|x\|_\infty;$$

- Low rank approximation

$$\text{Argmin}_x \|X - Y\|_F + \mu \|X\|_*;$$

the proximal operator of the nuclear norm is the soft-thresholding of the singular values;

- Phase retrieval.

Implementation details are important:

- To choose the step size, approximate f as a quadratic function $f(x) \approx \frac{1}{2}a \|x\|^2 + \langle x, b \rangle + x$, and set $\tau = 1/a$;
- Fista acceleration: after the proximal step, continue to move in the same direction;
- Backtracking line search: if the objective does not decrease (or if it increases more than some threshold, for non-monotone line search), half the step size;
- Stop when the relative residual

$$\frac{\|\nabla f + \nabla g\|}{\varepsilon + \max\{\|\nabla f\|, \|\nabla g\|\}} \text{ or } \frac{\|\nabla f + \nabla g\|}{\varepsilon + \|\nabla f(x_1) + \nabla g(x_1)\|}$$

is below some threshold.

The Frank-Wolfe algorithm

To minimize a convex function f on a bounded *polygonal* region:

- Start in x_k ;
- Linearize the objective f at x_k ;
- Solve the resulting linear program: c_k^* ;
- The direction $\overrightarrow{x_k x_k^*}$ is a descent direction: use a linear search to select the next point

$$x_{k+1} = \underset{x \in [x_k, x_k^*]}{\operatorname{Argmin}} f(x);$$

- Iterate.

No free lunch theorems for optimization **D.H. Wolpert and W.G. Macready (1997)**

Without any assumption on the function to be minimized, no algorithm can be guaranteed to be good.

Copositive programming **J. Matoušek and B. Gärtner**

Copositive matrices form a convex cone

$$X = \{M \text{ symmetric} : \forall x \geq 0 \quad x' M x \geq 0\}.$$

Its dual is the set of completely positive matrices

$$P = \{M : \exists x_i \geq 0 \quad M = \sum x_i x_i'\}.$$

Copositive programming is NP-hard.

Extending formulations in mixed-integer convex programming **M. Lubin et al.**

To solve mixed integer *convex* programs, prefer polyhedral (but still integral) relaxations to branch-and-bound non-linear programming (NLP): MILP solvers are fast.

Unit tests for stochastic optimization **T. Schaul et al.**

Build unit tests for stochastic optimization algorithms (variants of stochastic gradient descent):

- Start with prototypes , , , ,  (Gaussian),  (Laplace);

- Concatenate them;
- Add noise to the gradient (additive Gaussian or Laplace (for outliers), multiplicative, dropout);
- Compose them into higher-dimensional signals, e.g., with $(x, y) \mapsto (x^p + y^p)^{1/p}$, rotate;
- Add some curl (in reinforcement learning (RL), those algorithms are fed non-gradients);
- Let the function slowly change with time (also common in RL).

Implementation in `lua-torch`.

A tutorial on spectral clustering **U. von Luxburg (2007)**

Given a cloud of points, one can build similarity graphs (prefer connected ones):

- ε -neighbourhood graph (set ε to the length of the longest edge in the minimum spanning tree);
- k -nearest neighbour graph, $k = \log n$;
- Mutual k -nearest neighbour graph, $k > \log n$;
- Fully-connected graph, with Gaussian similarity,

$$s(x_i, x_j) = \exp - \frac{\|x_i - x_j\|^2}{2\sigma^2}$$

(set σ to ε , or to the mean distance to the k th nearest neighbour, $k = \log n$).

Given a graph, with weight and degree matrices W and D , one can consider its Laplacians (there are three of them):

- $L = D - W$;
- $L_{\text{sym}} = D^{-1/2} L D^{-1/2}$;
- $L_{\text{rw}} = D^{-1} L$.

Spectral clustering is k -means (or any other clustering algorithm) on the span of the first k eigenvectors of L , L_{sym} (requires normalization) or L_{rw} ; prefer L_{rw} .

The ratio and normalized cut problems, finding a graph partition that minimizes

$$\text{RatioCut}(A_1, \dots, A_n) = \frac{1}{2} \sum \frac{W(A_i, \bar{A}_i)}{|A_i|}$$

$$\text{or } \text{NCut}(A_1, \dots, A_n) = \frac{1}{2} \sum \frac{W(A_i, \bar{A}_i)}{\text{vol } A_i}$$

where $\text{vol } A = \sum_{i \in A} d_i$, are NP-hard, but their convex relaxations are equivalent to spectral clustering (for a partition $V = A \sqcup \bar{A}$, consider the vector $f_i = \sqrt{|\bar{A}| / |A|}$ or $\sqrt{\text{vol } \bar{A} / \text{vol } A}$ if $i \in A$, and the inverse otherwise; notice that

$$f' L f = |V| \cdot \text{RatioCut}(A, \bar{A}) \quad \text{and } f \perp \mathbf{1}$$

$$\text{or } f' L f = \text{vol}(V) \cdot \text{RatioCut}(A, \bar{A}) \quad \text{and } D f \perp \mathbf{1}.$$

Both L and L_{rw} promote dissimilarity between clusters (good cuts), but L_{rw} also promotes similarity within clusters.

The transition matrix of a random walk on a graph is $P = D^{-1} W = I - L_{\text{rw}}$; random walks are related to cuts: $\text{NCut}(A, \bar{A}) = P(\bar{A}|A) + P(A|\bar{A})$.

The *commute distance* c_{ij} is the expected time for the random walk to go from i to j and back,

$$c_{ij} = \text{vol}(V)(e_i - e_j)'L^\dagger(e_i - e_j);$$

therefore $\sqrt{c_{ij}}$ can be used as a Euclidian distance.

The commute time embedding and the unnormalized spectral embedding (via L) are related but different; for connected graphs, they are very similar.

The graph Laplacian is close to that of a graph in which the communities are disconnected; therefore, the eigenvalues and eigenvectors are also close (Davis-Kahan theorem – how close depends on the *spectral gap*). This is more useful for L , L_{rw} than for L_{sym} , S (similarity matrix) or W (weights), because the eigenvalues are ordered, bounded away from zero, and (contrary to L_{sym} , the eigenvectors are unscaled.

The Laplacian spectrum of graphs **B. Mohar (1991)**

The Laplacian of a graph is $Q = \text{diag}(D) - A$, where A is the adjacency matrix and $D = A\mathbf{1}$ is its degree vector;

$$\langle Qx, x \rangle = \sum_{(u,v) \in E} a_{uv}(x_u - x_v)^2.$$

Equivalently, $Q = CC'$, where C is the oriented incident $|V| \times |E|$ -matrix with entries

$$c_{ve} = \begin{cases} +1 & \text{if } e = * \rightarrow v \\ -1 & \text{if } e = v \rightarrow * \\ 0 & \text{otherwise.} \end{cases}$$

On a lattice, C is a discretization of the gradient, Q of the Laplacian.

Bounds on the eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$ are known. The second eigenvalue,

$$\lambda_2 = \min_{\substack{x \perp \mathbf{1} \\ x \neq 0}} \frac{\langle Qx, x \rangle}{\langle x, x \rangle}$$

is a measure of connectivity. It is related to (appears in inequalities involving) expander graphs, diameter, number of edges in a maximal cut, isometric number, mean distance, independence number, genus, vertex connectivity, edge connectivity, etc.

The eigenvectors y for λ_2 can be used for 1-dimensional dimension reduction (heuristically, they reduce the length of the jumps $|i - j|$ for edges $i - j$); $\{v : y_v > c\}$ and $\{v : y_v < c\}$ are connected (if $\forall v y_v \neq c$).

The number of spanning trees in G is the coefficient of x in the characteristic polynomial of its Laplacian.

Spectral clustering of graphs **with the Bethe Hessian** **A. Saade et al. (2014)**

The *non-backtracking matrix* of a graph is the (symmetric) binary matrix, indexed by edges, identifying

patterns of the form $i \rightarrow j \rightarrow k$ with $i \neq k$: $B_{i \rightarrow j, \ell \rightarrow k} = \delta_{j\ell}(1 - \delta_{ik})$.

The *Bethe Hessian*, a smaller, symmetric matrix, is a deformation of the Laplacian, $H(r) = (r^2 - 1)\mathbf{1} - rA + D$.

The eigenvalues of B are the roots of $\det H(r)$. The eigenspaces of B , and $H(r)$, for $r = \rho(B)$ (the spectral radius of B can be approximated as $\langle d^2 \rangle / \langle d \rangle - 1$; the approximation can then be refined by looking for the closest root of $\det H(r)$) can help separate communities.

An overview of low-rank matrix recovery **from incomplete observations** **M.A. Davenport and J. Romberg (2016)**

Low-rank matrix recovery can be approximated as follows.

Minimize $_X \|X - X_0\|_F^2$ subject to $\text{rank } X = r$ (SVD)

Minimize $_X \|X - X_0\|_F^2 + \lambda \|X\|_*$

Minimize $_X \|AX - y\|_2^2 + \lambda \|X\|_*$

Minimize $_{L,R} \|ALR' - y\|_2^2 + \frac{1}{2}\lambda \|L\|_* + \frac{1}{2}\lambda \|R\|_*$

Minimize $_X \|X\|_*$ subject to $AX = y$

One can use *iterative hard thresholding* (IHT, projected gradient: move in the direction of $\nabla \|AX - y\|_2^2$, use the SVD to reduce the rank to r , iterate) or alternating projections

$$R \leftarrow \underset{R}{\text{Argmin}} \|ALR' - y\|_2^2$$

$$L \leftarrow \underset{L}{\text{Argmin}} \|ALR' - y\|_2^2$$

Instead of the nuclear norm $\|\cdot\|_*$, one can consider the max-norm, or the log-determinant (it is not convex, but it is a smooth rank proxy for positive semi-definite matrices).

Solving systems of quadratic equations is equivalent to finding a rank-1 matrix $X = vv'$ satisfying linear constraints $\langle X, A_i \rangle = b_i$ (*lifting*)

$$\text{Minimize}_X \|X\|_* \text{ subject to } \forall i \langle X, A_i \rangle = b_i.$$

Applications include *phase retrieval*: recovering v from the magnitudes $|\langle v, a_i \rangle| = y_i$,

$$\text{Minimize}_X \|X\|_* \text{ subject to } \forall i \langle X, a_i a_i' \rangle = y_i^2.$$

Compressive PCA **for low-rank matrices on graphs** **N. Shahid et al.**

Given a matrix $Y \in \mathbf{R}^{p \times n}$, build the k -nearest graph on its columns (resp. rows), with Gaussian weights, compute its Laplacian and its eigenvalue decomposition. The matrix Y is (k_r, k_c) -low rank on graphs if its columns (resp. rows) are in the span of the first k_c (resp. k_r) eigenvectors. (For big data, sample the rows and columns of Y .) Low rank matrices on graphs capture non-linear low-rank structures.

***Pursuits in structured
non-convex matrix factorizations***
R. Khanna et al.

At each step, greedily add the best rank-1 matrix for the linearized objective.

Riemannian pursuit for big matrix recovery
M. Tan et al. (2014)

The matrix lasso, for low rank matrix reconstruction

$$X = \underset{X}{\operatorname{Argmin}} \frac{1}{2} \|b - \mathcal{A}X\|_2^2 + \lambda \|X\|_*$$

does not scale well, because of the high-dimensional SVD needed to compute the nuclear norm $\|X\|_*$. Instead, one can use *fixed rank* methods: rank r matrices form a Riemannian manifold

$$\begin{aligned} \mathcal{M}_n &= \{X \in \mathbf{R}^{n \times n} : \operatorname{rank} X = r\} \\ &= \{U \operatorname{diag}(\sigma) V' : U \in \operatorname{St}_r^n, V \in \operatorname{St}_r^n, \|\sigma\|_0 = r\} \\ \operatorname{St}_r^m &= \{U \in \mathbf{R}^{m \times r} : U'U = I\} \quad (\text{Stiefel manifold}). \end{aligned}$$

The rank r is unknown: progressively increase it.

Convex banding of the covariance matrix
J. Bien et al. (2014)

Banded covariance matrices can be recovered (if the order on the variables is known) with a *hierarchical group lasso* penalty (sum of weighted unsquared L^2 norms – the weights are needed to avoid overpenalizing overlapping regions).

***3D shape reconstruction from 2D landmarks:
a convex formulation***
X. Zhou et al. (2014)

To reconstruct a 3D shape from a set of 2D landmarks W , assume that the shape is an unknown linear combination of simpler shapes $W = \Pi(R \sum c_i B_i + T)$, where Π is the camera matrix, R the rotation, T a translation, and B_i the simpler shapes. This can be simplified to $W = \bar{R} \sum c_i B_i$, $\bar{R} \bar{R}' = I$.

Since the convex hull of the Stiefel manifold $\{X : X'X = I\}$ is the unit (spectral norm) ball, one can consider the convex relaxation

$$\underset{R_i}{\operatorname{Argmin}} \left\| W - \sum R_i B_i \right\|_F^2 + \lambda \sum \|R_i\|_2.$$

***Multi-scale structure and topological anomaly
detection via a new network statistic:
the onion decomposition***
L. Hébert-Dufresne et al.

To build the k -core decomposition of a network:

- Remove isolated nodes;
- Remove nodes connected to 1 (later, k) fewer nodes; iterate (because this lowers the degree of the remaining nodes);

- Increase k and iterate.

The *onion decomposition* keeps track of the nodes in each inner iteration. The *onion spectrum* is the number of nodes in each layer of the onion decomposition; various micro/meso/macro characteristics are visible on it (assortativity, loops/trees, core/periphery, centrality).

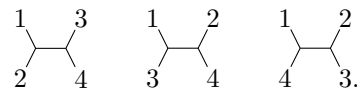
It is easy to generate random graphs with prescribed onion spectrum, and they have the same characteristics (e.g., shortest path length distribution) as real-world networks.

***Statistical models for cores decomposition
of an undirected random graph***
V. Karwa et al.

The k -core of a graph is the maximal subgraph in which every vertex has degree at least k . The *shell index* of a node indicates the highest core in which it is. Erdős-Rényi random graphs specify the degree distribution, but the degree is only a coarse measure of centrality: instead, one can use the shell index distribution.

***A fast quartet tree heuristic
for hierarchical clustering***
R. Cilibrasi and M.B. Vitányi (2014)

In a tree, each subset of 4 nodes has one of 3 possible layouts (“quartet topologies”),



Randomized hill-climbing (C implementation in the CompLearn library) can find the tree maximizing the number of consistent quartet topologies.

***Statistically-consistent k-mer methods
for phylogenetic tree reconstruction***
E.S. Allman et al.

Hierarchical clustering of sequences from n -gram Euclidean distance can be inconsistent: prefer model-based distances.

***Scalable force-directed graph layout algorithms
using fast multipole methods***
E. Yunis et al.

Barnes-Hut is $O(n \log n)$ and considers particle-cluster interactions; the fast multipole method (FMM) considers cluster-cluster interactions to reduce the computations and is $O(n)$. The ExaFMM implementation can also be used for graph layout.

***Efficient and high-quality
force-directed graph drawing***
Y. Hu

Force-directed graph-drawing algorithms include:

- Fruchterman-Reingold: edges are springs; there is a repulsive force for each pair of nodes;

- Kamada-Kawai: springs between all pairs of vertices, with length proportional to the distance in the graph;
- Walshaw: to avoid local minima, start with a coarse graph and progressively refine it; only consider forces from neighbouring vertices;
- Combine this multilevel approach with Barnes-Hut oct-trees and progressive cooling.

***The hidden hyperbolic geometry
of international trade:
world trade atlas 1870-2013***
G. García-Pérez et al. (2016)

Plot the world trade map in the hyperbolic plane (Poincaré disk), not the Euclidian one.

Kriging of financial term structures
A. Cousin et al. (2016)

Adding monotonicity constraints to a Gaussian process makes it non-Gaussian but, after discretization, it remains tractable.

***Trading networks, abnormal motifs
and stock manipulation***
Z.Q. Jiang et al. (2012)

The order book has the (encrypted) identity of the trader: look for abnormal motifs such as \mathcal{Q} (self-buy), \rightleftarrows (stock pool) or \rightleftarrows (preferred counterparty).

***Measuring nonlinear dependence
in time series, a distance correlation approach***
Z. Zhou (2014)

The *distance correlation* can be applied to time series

$$V(X, Y) \propto \iint \frac{f_X(x)f_Y(y)}{|x|^{p+1}|y|^{p+1}} dx dy$$

$$V_k(X) = V(X_n, X_{n+k})$$

$$R_k(X) = \sqrt{\frac{V_k(X)}{V_0(X)}}$$

where f_X is the characteristic function of X .

It can be estimated as

$$\hat{V}_k(X) \propto \sum A_{r\ell} B_{r\ell}$$

$$A_{r\ell} = a_{r\ell} - a_{r\bullet} - a_{\bullet\ell} + a_{\bullet\bullet}$$

$$a_{r\ell} = |x_r - x_\ell|.$$

The distance correlation tests need to be adjusted for dependence.

***Clustering of time series subsequences
is meaningless:
implication for previous and future research***
E. Keogh and J. Lin

Before clustering 1-dimensional time series, one may be tempted to embed them in a d -dimensional space (using a sliding window): the resulting clusters do not depend on the dataset; they are just an artefact of the embedding (the cluster centers are translation-invariant and turn out to be (approximately) sine waves).

***The ladder: a reliable leaderboard
for machine learning competitions***
M. Hardt and A. Blum (2015)

Compare the new loss with the best loss so far, with a 1-sided paired T -test, and only update the performance if it is significantly better (you need to remember the whole loss vector).

***How to scale up kernel methods
to be as good as deep neural nets***
Z. Lu et al. (2014)

Kernel-based algorithms, e.g., support vector machines (SVM), tend to scale quadratically with the number of observations, making them unsuitable for large datasets.

Translation-invariant continuous positive definite kernels, $k(x, y) = k(x - y)$, transform the Dirac measure into the Fourier transform of a non-degenerate probability density function, $k(\delta) = \mathcal{F}p$, i.e.,

$$k(x - y) = \int p(\omega) e^{i\omega'(x-y)} d\omega = E_\omega [e^{i\omega'x} e^{-i\omega'y}].$$

They can be approximated by random features

$$k(x - y) \approx \frac{1}{D} \sum_{i=1}^D \phi_{\omega_i}(x) \phi_{\omega_i}(y)$$

$$\phi_{\omega}(x) = \cos(\omega'x + b_i)$$

$$\omega \sim N(0, \sigma^{-1}I) \quad \text{for a rbf kernel}$$

$$\omega \sim \text{Cauchy}(\sigma) \quad \text{for a Laplace kernel}$$

$$b_i \sim U(0, 2\pi)$$

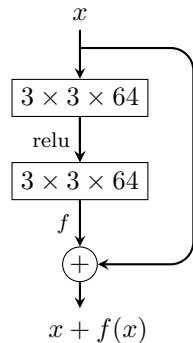
Computations can be parallelized: train a model on a subset of 25,000 random features; combine the resulting models, e.g., by averaging the log-probabilities of the classes.

Those kernels can be added (multiple kernel learning, MKL: either concatenate the random features of all the models, or fit the models separately and combine the forecasts with non-negative weights), multiplied (just add the corresponding ω_i 's) and even sometimes composed – the resulting kernel is then equivalent to a neural net, with one hidden layer and random weights (echo state network, extreme learning machine, reservoir learning, etc.).

Deep residual learning for image recognition

K. He et al. (2015)

The skip-layer connections in a residual net lead to additive (rather than multiplicative) gradient back-propagation, thereby avoiding vanishing or exploding gradients.

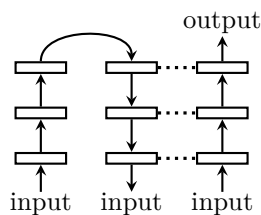


They are also easy to initialize: $f = 0$ or f small.

Deconstructing the ladder network architecture

M. Pezeshki et al. (2016)

Ladder networks, for semi-supervised learning, combine a denoising auto-encoder (for the unsupervised part) with a neural net (for the supervised part), with the decoder and the neural net constrained (or penalized) to share weights.



Semi-supervised learning with ladder networks

A. Rasmus et al. (2016)

A ladder network is a (denoising) auto-encoder that tries to reconstruct, not only the input, but all the intermediate reconstructions.

Binary embeddings with structured hashed projections

A. Choromanska et al.

Pseudo-random projections, with a structured matrix (circulant, Toeplitz, etc.)

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & & & \\ \vdots & \diagdown & & x_2 \\ x_n & \cdots & x_2 & x_1 \end{pmatrix} \quad \begin{pmatrix} x_0 & x_1 & \cdots & \\ x_{-1} & & & \\ \vdots & \diagdown & & x_1 \\ \vdots & & x_{-1} & x_0 \end{pmatrix}$$

followed by the sign function, for dimension reduction (lossy compression), e.g., before feeding data (MNIST) to a neural net.

Are elephants bigger than butterflies?

Reasoning about sizes of objects

H. Bahgerinezhad et al.

To estimate object sizes from images:

- Observe the (depth-adjusted) ratio of bounding box areas, in Flickr images tagged with two objects;
- Arrange those ratios into a graph (not all pairs are observed, and some are noisy);
- Model the sizes as log-normal distributions.

Asynchronous methods for deep reinforcement learning

V. Mnih et al. (2016)

Asynchronously running multiple agents in parallel makes deep reinforcement learning amenable to multicore CPUs instead of GPUs.

The Libra toolkit for probabilistic models

D. Lowd and A. Rooshenas (2015)

Command-line tool to estimate and use graphical models; also check libDAI, OpenGML, BNT, OpenMarkov FastInf.

Tanh-Sinh high-precision quadrature

D.H. Bailey (2006)

To approximate integrals, use

$$\int_{-1}^1 f = \int_{-\infty}^{+\infty} f(g(t))g'(t)dt \approx \sum_{j=-N}^N g'(hj)f(g(hj))$$

with $h = 2^{-12}$ for 1000-digit precision,

$$\text{error} \approx \frac{h^2}{(2\pi)^2} \sum F''(jh)$$

$$F(t) = f(g(t))g'(t)$$

$$g(t) = \tanh\left(\frac{\pi}{2} \sinh t\right).$$

Laplace deconvolution with noisy observations

F. Abramovich et al.

The Laplace deconvolution problem is the task of recovering a function f from a kernel g and

$$q(t) = \int_0^t g(t-s)f(s)ds. \quad (\text{Volterra equation})$$

It could be solved using the Laplace transform but, in practice, the observations of q are discrete and noisy; one can, for instance, expand f in the basis of Laguerre functions ($e^{-t} \times$ Laguerre polynomials).

E pur si muove: Galilean-invariant cosmological hydrodynamical simulations on a moving mesh
V. Springel (2009)

Dynamic (moving, unstructured, Voronoi) meshes, which move with the flow can be used for hydrodynamic cosmological simulations.

To compute a Delaunay triangulation: start with a valid triangulation, add a point, add edges to the vertices of its containing triangle, check the in-circle property for the edges of all the triangles created, flip them if needed.

To regularize a Voronoi tessellation: start with a Poisson sample (or a non-uniform distribution) with periodic boundary conditions, move the points to the center of mass of their cells, iterate 150 times).

Refine/de-refine the mesh whenever/wherever needed.

A linguistic approach to categorical color assignment for data visualization
V. Setlur and M.C. Stone (2015)

Assigning meaningful colours to categories, using n -gram co-occurrence (to assess colourability and infer the main colours) and Google image search and WordNet (to find actual, precise colours, from clipart images).

A survey of predictive modelling under imbalanced distributions
P. Branco et al. (2015)

Evaluating the performance of a classifier or a regression algorithm on unbalanced data requires specific metrics, e.g.:

- F_1 : harmonic mean of recall and precision;
- Geometric mean of sensitivity and specificity;
- Area under the ROC (true positive rate vs false positive rate) curve;
- Area under the precision-recall curve;
- Index of balanced accuracy, $(1 + \alpha \cdot \text{dominance}) \cdot M$, where M is some performance measure, and the dominance is the difference between true positive and negative rates.

There are regression analogues of the ROC curve (for those, you often want the area *over* the curve, AOC):

- RROC: $-\sum(\hat{y}_i - y_i + c)_- \sim \sum(\hat{y}_i - y_i + c)_+$;
- REC curve: $P[|y - \hat{y}| < \varepsilon] \sim \varepsilon$;
- REC surface: probability $\sim t + \text{error}$.

To deal with unbalanced datasets, try:

- Random undersampling of the majority class;
- Random oversampling of the minority class or, better, synthetic data (SMOTE: synthetic minority over-sampling technique);
- Reweighting (not unlike boosting);
- Identify **Tomek links**, *i.e.*, points of different classes that are each other's nearest neighbours, and remove that in the majority (or both);

- **Condensed nearest neighbour** (CNN): find a subset of the data that correctly classifies the training set with 1-nearest neighbours;
- 1-class SVM, auto-encoders;
- Utility-based optimization, e.g., from a cost-benefit matrix (if available).

Check the **unbalanced** (R) or **imbalance-learn** (Python) packages.

Related problems include overlapping, small datasets, high dimension, noise, small disjuncts (the minority class has clusters of different sizes or densities: it is easy to miss the smallest ones).

Are random forests truly the best classifiers?
M. Wainberg et al. (2016)

When playing with hyperparameters, you must use a held-out test set.

Estimating uncertainty for massive data streams
N. Chamandy et al. (2012)

The **Poisson bootstrap** (for streaming data, *i.e.*, when the number of observations is not known), uses weight $\sim \text{Pois}(1)$.

A tutorial introduction to the minimum description length principle
P. Grünwald (2004)

The minimum description length (MDL) principle is an information-theoretic interpretation of penalized likelihood estimators. It finds hypotheses that compress the data the most, often using a decomposition

$$\underset{H \in \mathcal{H}}{\text{Argmin}} L(H) + L(D|H)$$

into model length $L(D)$ (e.g., the order of a Markov chain and its coefficients) and residual length $L(\text{Data}|H)$. The first term (penalty, model complexity) identifies structure in the data, the second (log-likelihood), measures the residual noise.

A probability distribution P on a finite set $\mathcal{Z} = \mathcal{X}^n$ determines an optimal code C with code length $L(z) = \lceil -\log_2 P(z) \rceil$. If $n \gg 1$, the rounding is negligible. The actual code is not needed, only the *code length*, *i.e.*, a function $L : \mathcal{Z} \rightarrow [0, \infty]$ such that $\sum_z 2^{-L(z)} \leq 1$. Probability and code length are related by

$$L = \underset{L}{\text{Argmin}} E_{Z \sim P}[L(Z)]$$

$$L(z) = -\log_2 P(z).$$

(This can be generalized to continuous distributions: $L(z)$ is then the length of the code needed to encode $z \in \mathcal{X}^n$ with “unit precision”.)

The information inequality is

$$E_P[-\log Q(Z)] \geq E_P[-\log P(Z)].$$

The crude, 2-part MDL principle is

$$H = \underset{H}{\operatorname{Argmin}} L(H) + L(D|H),$$

where $L(D|H) = \log_2 P(D|H)$ and $L(H)$ is defined in an ad hoc fashion (for instance, to model a sequence of length n using Markov chains, one could first encode the order of the chain and then the counts – this is finite, and is a sufficient statistic).

Instead of $L(D|H)$, we do not lose much if we use $L(D|\mathcal{H})$: if \mathcal{H} is finite, $\mathcal{H} = \{H_1, \dots, H_k\}$, find the hypotheses H_i with the smallest code length $L(D|H_i)$, consider its code C_i , and encode the data D as $(i, C_i(D))$ (2-part *universal model*). This still works if \mathcal{H} is countably infinite. More generally (this will also work for continuous models), for $\mathcal{H} = \{P(\cdot|\theta), \theta \in \Theta\}$, consider a prior W on Θ and the *Bayesian mixture* $P(\cdot) = \sum_{\theta \in \Theta} P(\cdot|\theta)W(\theta)$. The corresponding code length gives the Bayesian universal model.

The normalized maximum likelihood (NML, aka Shtarkov) distribution

$$P(x^n) \propto P(x^n | H_{\hat{\theta}(x^n)}),$$

which minimizes the worst-case regret,

$$\max_{x^n \in \mathcal{X}^n} L(x^n|P) - L(x^n|P_{\hat{\theta}(x^n)})$$

gives the minimax optimal (NML) model.

The size distribution of inhabited planets **F. Simpson (2016)**

Over 98% of the population lives in a country larger than the median: by analogy, most inhabited planets are smaller than ours, and have fewer and heavier (300 kg) inhabitants.

Memcomputing NP-complete problems in polynomial time using polynomial resources **F.L. Traversa et al. (2014)**

A memcomputer is a non-Turing machine, equivalent to a non-deterministic Turing machine (it can solve NP-complete problems with polynomial resources, even if $P \neq NP$) whose units can both store and process information – they not only store bits but waveforms, e.g., combining $(e^{i\omega_k t})_{1 \leq k \leq n}$ into $\prod_k e^{i\omega_k t}$.

NP-complete problems and physical reality **S. Aaronson**

To solve NP-complete problems in polynomial time, find a physical phenomenon whose description is equivalent to an NP problem, and just observe it.

Dirichlet processes **Y.W. Teh**

The *Dirichlet distribution* is conjugate to the multinomial; it generalizes the Beta distribution (conjugate to

the Bernoulli) from the interval $[0, 1]$ to the simplex Δ_k .

Bayesian density estimation considers the model

$$F \sim \text{Prior}$$

$$x_i \sim F$$

and computes the posterior distribution $F|x_1, \dots, x_n$.

The *Bayesian mixture model* with k components can be written

k	number of components
$\alpha > 0$	strength of the prior
$\pi \sim \text{Dir}\left(\frac{\alpha}{k}, \dots, \frac{\alpha}{k}\right)$	cluster probabilities
$z_i \sim \text{Mult}(\pi)$	cluster membership
$\theta_k \sim H$	cluster parameters
$x_i \sim F_{\theta_{z_i}}$	

For instance, in the Gaussian case, H is the normal-inverse-Wishart distribution, $\theta_k = (\mu_k, V_k)$ and $F_{\theta_k} = N(\mu_k, V_k)$. The *Dirichlet process* (DP) appears when we let $k \rightarrow \infty$ (leaving the number of observations fixed).

There are many ways of defining, constructing or sampling from a Dirichlet process $G \sim \text{DP}(\alpha, H)$. Let Θ be a measurable space, e.g., $[0, 1]$ or $\mathbf{R}^n \times \text{SPD}_n$, H a probability distribution on Θ , e.g., uniform on $[0, 1]$ or Normal-inverse-Wishart on $\mathbf{R}^n \times \text{SPD}_n$, and $\alpha > 0$ the strength of the prior.

A Dirichlet process is a family of random variables $(G(A))_{A \subseteq \Theta}$ whose finite margins are Dirichlet: for all (measurable) partition $\Theta = A_1 \sqcup \dots \sqcup A_n$,

$$(G(A_1), \dots, G(A_n)) \sim \text{Dir}(\alpha H(A_1), \dots, \alpha H(A_n)).$$

A **Dirichlet process** is a random probability distribution G on Θ whose samples are of the form $\sum_{k \geq 1} \pi_k \delta_{\theta_k}$ with (broken stick construction)

$$\begin{aligned} \theta_k &\sim H \\ \pi_k &= \beta_k \prod_{1 \leq \ell \leq k} (1 - \beta_\ell) \\ \beta_k &\sim \text{Beta}(1, \alpha). \end{aligned}$$

This is sometimes written $\pi \sim \text{GEM}(\alpha)$.

The *urn scheme* explains how to sample from a Dirichlet process $G \sim \text{DP}(\alpha, H)$, sample from that sample, $\theta_1, \dots, \theta_n \sim G$, and marginalize G out. Start with a sample $\theta_1 \sim H$. Once you have $\theta_1, \dots, \theta_n$, with probability $\alpha/(\alpha + n)$, sample a new point $\theta_{n+1} \sim H$; with probability $n/(\alpha + n)$, sample from the existing points $\theta_{n+1} \sim \text{Unif}(\{\theta_1, \dots, \theta_n\})$.

The *Chinese restaurant process* (CRP) is similar, but only looks at the partition of $\llbracket 1, n \rrbracket$ induced by the unique values in $(\theta_1, \dots, \theta_n)$.

The **Dirichlet process mixture** model (or infinite mixture model) can be written

Θ	parameter space
H	prior
α	strength of the prior
$G \sim \text{DP}(\alpha, H)$	
$\theta_i \sim G$	
$x_i \sim F_{\theta_i}$	

(note that there are usually duplicated values in the θ_i) or, equivalently

Θ	parameter space
H	prior
α	strength of the prior
$\theta_k \sim H$	cluster k parameters
$\pi \sim \text{GEM}(\alpha)$	cluster probabilities
$z_i \sim \text{Mult}(\pi)$	cluster membership
$x_i \sim F_{\theta_{z_i}}$	i th observation

Hierarchical Bayesian nonparametric models with applications

Y.W. Teh and M.I. Jordan (2009)

A **hierarchical Dirichlet process** (HDP) is a Dirichlet process (DP) whose base distribution is a sample from a DP. For instance, the *HDP mixture model* allows multiple clustering problems to share clusters, e.g., to share topics across documents.

clustering problem cluster	corpus	population
	document	genetic makeup
	topic	subpopulation

The infinite HMM (iHMM) or HDP-HMM is a hidden Markov model (HMM) with an unbounded number of hidden states (one needs to increase the probability of self-transitions to avoid the creation of many redundant states).

The **Pitman-Yor process**, which replaces $\text{Beta}(1, \alpha)$ with $\text{Beta}(1 - d, \alpha + kd)$ and yields the power laws commonly seen in language modeling, also has a hierarchical variant (HPY).

A random measure G is *completely random* if

A_1, \dots, A_n disjoint $\implies G(A_1), \dots, G(A_n)$ independent.

The Dirichlet process is not completely random because it is a probability measure. The *Beta process* is a completely random measure generalizing the Dirichlet process and often used as a prior for *featural representations* (i.e., binary matrices, i.e., clustering problems where an observation can be in several clusters). A sample from a Beta process $B \sim \text{BP}(c, B_0)$ is of the form $B = \sum_{k \geq 1} \omega_k \delta_{\theta_k}$, where $(\omega_k, \theta_k)_{k \geq 1}$ is a sample from a Poisson process on $[0, 1] \times \Theta$ with rate measure $\nu(d\omega, d\theta) = c\omega^{-1}(1 - \omega)^{c-1}d\omega B_0(d\theta)$ where $c > 0$ (concentration).

There is a stick-breaking construction (for $c = 1$)

$$B = \sum \omega_k \delta_{\theta_k}$$

$$v_k \sim \text{Beta}(1, \alpha)$$

$$\omega_k = \prod_{i=1}^k (1 - v_i)$$

$$\theta_k \sim B_0$$

or (2-parameter Beta process)

$$B = \sum_{n \geq 1} \sum_{k=1}^{K_n} \omega_{nk} \delta_{\theta_{nk}}$$

$$K_n \sim \text{Poisson}\left(\frac{c\alpha}{c + n - 1}\right)$$

$$\theta_{nk} \sim B$$

$$\omega_{nk} \sim \text{Beta}(1, c + n - 1)$$

Bayesian nonparametric models

P. Orbanz and Y.W. Teh

A nonparametric Bayesian model is an infinite-dimensional (parametric) Bayesian model, i.e., a finite-dimensional parametric Bayesian model whose dimension increases with sample size. Examples include:

– Gaussian process: $f \sim \text{GP}$ means

$$(f(x_1), \dots, f(x_n)) \sim N;$$

- Dirichlet process $G = \sum_{k \geq 1} \pi_k \delta_{\theta_k}$, $\sum \pi_k = 1$;
- Chinese restaurant process (partitions from a DP);
- Beta process $G = \sum_{k \geq 1} \pi_k \delta_{\theta_k}$;
- Indian buffet process (partitions from a BP);
- Pitman-Yor process;
- Hierarchical DP, BP or PY;
- Dependent DP;
- etc.

Showing that those processes can be tricky:

- Compatible finite-dimensional marginals and Kolmogorov's extension theorem (only for countable families);
- Explicit construction (e.g., stick-breaking);
- Limit of finite-dimensional distributions;
- Exchangeable sequences and de Finetti's theorem.

Consistency of infinite-dimensional Bayesian models is not as common as for finite-dimensional ones.

In R, check the **DPpackage** package.

Natural language processing

M. Collins (Coursera & Columbia, 2013)

1. A language model is a probability distribution on V^+ , the set of sentences written with a vocabulary V . The sample distribution, from a corpus, is not a good model: it assigns a zero probability to sentences never observed. Markov models (**n -gram** models) assume that $P(x_{k+1}|\text{past}) = P(x_{k+1}|x_{1:k}) = P(x_{k+1}|x_{k-n+1:k})$. These probabilities can be estimated as follows:

$$\begin{aligned}
- P(x_x|x_1, x_2) &= \frac{\text{count}(x_1, x_2, x_3)}{\text{count}(x_1, x_2)} \\
- P(x_3|x_1, x_2) &= \lambda_1 \frac{\text{count}(x_1, x_2, x_3)}{\text{count}(x_1, x_2)} + \\
&\quad \lambda_2 \frac{\text{count}(x_2, x_3)}{\text{count}(x_2)} + \\
&\quad \lambda_3 \frac{\text{count}(x_3)}{\# \text{ words}}
\end{aligned}$$

where $\lambda_1, \lambda_2, \lambda_3$ depend on the trigram (x_1, x_2, x_3) , (in particular, $\lambda = 0$ if the denominator is zero), e.g.,

$$\begin{aligned}
\lambda_1 &= \frac{\text{count}(x_1, x_2)}{\text{count}(x_1, x_2) + \gamma} \\
\lambda_1 &= (1 - \lambda_1) \frac{\text{count}(x_2)}{\text{count}(x_2) + \gamma} \\
\lambda_3 &= 1 - \lambda_1 - \lambda_2
\end{aligned}$$

$$- P(x_2|x_1) = \frac{\text{count}^*(x_1, x_2)}{\text{count}(x_1)} \text{ if } \text{count}(x_1, x_2) > 0,$$

where the discounted count is

$$\text{count}^*(x_1, x_2) = \text{count}(x_1, x_2) - \beta$$

and the missing probability mass

$$1 - \sum_{x_2} \frac{\text{count}^*(x_1, x_2)}{\text{count}(x_1)}$$

is assigned to the bigrams with no counts.

2. One can hope to describe sentences with a *context-free grammar* (CFG). For a given (grammatical) sentence, there can be several (left-most) derivations – the sentence can be ambiguous. A **probabilistic CFG** (PCFG) assigns a probability to each rule of the grammar, and therefore defines a probability distribution on the set of derivations (not just the set of sentences). The training data provides the grammar and empirical probabilities from the rules. The *CKY* algorithm uses dynamic programming to find the most likely parse tree (for a grammar in *Chomsky normal form* (CNF), i.e., whose rules are of the form $X \rightarrow Y_1 Y_2$ or $X \rightarrow y$) by computing $\pi(i, j, X) = \text{Max likelihood}(X \rightarrow x_{i:j})$. The *inside algorithm* similarly computes the probability of a sentence using $\pi(i, j, X) = \sum \text{likelihood}(X \rightarrow x_{i:j})$.

3. PCFGs are not sensitive to lexical information: for instance, they cannot notice that **into/IN** (resp. **of/IN**) is more (less) likely to be attached to a VP than to a NP. There is also no way of encoding the “close attachment preference” (a PP is more likely to be attached to a nearby NP than to a distant one). A **lexicalized PCFG** is a PCFG in which each non-terminal is paired with a lexical item (the “head” of the non-terminal), e.g.,

$$S(\text{examined}) \rightarrow_2 NP(\text{lawyer}) VP(\text{examined})$$

(the subscript indicates the origin of the lexical item). Since the number of parameters explodes, they should be regularized, for instance:

$$\begin{aligned}
- P[S(\text{examined}) \rightarrow_2 NP(\text{lawyer}) VP(\text{examined})] &= \\
P[S(\text{examined}) \rightarrow_2 NP VP(\text{examined})] P[\text{lawyer} | \dots]
\end{aligned}$$




- Shrink $P[S(\text{examined}) \rightarrow_2 NP VP(\text{examined})]$ towards $P[S \rightarrow_2 NP VP]$
- Shrink $P[\text{lawyer} | S(\text{examined}) \rightarrow_2 NP VP(\text{examined})]$ towards $P[\text{lawyer} | S \rightarrow_2 NP VP]$.

4. The **noisy channel** model (generative model) to translate from French f to English e is

$$p(e, f) = p(e)p(f|e).$$

It uses a language model for the target, $p(e)$, and a conditional model $p(f|e)$, even though we are eventually interested in $p(e|f)$.

It is too complicated to model $p(f_1 \dots f_n | e_1 \dots e_n, m)$ directly, but $p(f_1 \dots f_n, a_1 \dots a_m | e_1 \dots e_n, m)$, where $a_i \in \llbracket 1, n \rrbracket$ is the position of the English word corresponding to the i th French word, and the a_i can be integrated out. One can use a model for the alignments $P[a_i = j | m, n]$,

	similar languages
	noun/adjective inversions
	SVO vs SOV

another for the word pairings, $[f_i | e_j, a_i = j]$ and multiply them (IBM model 2).

The decoding problem, $\text{Argmax}_e p(e)p(f|e)$, is hard, but the alignment and word pairing models are easier, and useful in other models: use the EM algorithm (do not use hard alignments but weighted mixtures) and start with the uniform alignment probability (IBM model 1).

5. Considering phrases, i.e., consecutive groups of words, improves performance. To translate with a phrase model and a trigram language model, progressively build the translation, keeping track of the “state” (last two words (for the trigram model), last position (for the alignment model), boolean vector indicating which words have already been translated, score). Do not use a completely greedy algorithm, but keep a pool of good partial translations (states) and progressively expand them.

6. (Penalized) **log-linear models** generalize (smoothed) n -gram models: define features, e.g.,

$$\begin{aligned}
f_k(x_1, \dots, x_n) &= \mathbf{1}_{x_n = \text{world}} \\
f_k(x_1, \dots, x_n) &= \mathbf{1}_{x_{n-1}, x_n = \text{hello, world}}
\end{aligned}$$

let

$$p(x_{n+1} | x_1 \dots x_n) \propto \exp[\theta \cdot f(x_1 \dots x_n x_{n+1})]$$

or, writing $y = x_{n+1}$ and $x = (x_1, \dots, x_n)$,

$$\begin{aligned}
p(y|x) &\propto \exp[\theta \cdot f(x, y)] \\
p(y|x) &= \frac{\exp[\theta \cdot f(x, y)]}{\sum_z \exp[\theta \cdot f(x, z)]}
\end{aligned}$$

and find θ that maximizes the penalized log-likelihood

$$L(\theta) = \sum_i \log p(y^{(i)} | x^{(i)}, \theta) = \frac{1}{2} \|\theta\|_2^2$$

using gradient ascent – the gradient is easy to compute,

$$\frac{\partial L}{\partial \theta_k} = \sum_i f_k(x^{(i)}, y^{(i)}) - \sum_i \sum_z p(z|x^{(i)}) f_k(x^{(i)}, z).$$

A **log-linear model** to label a sentence is of the form

$$P[\text{label}|\text{input}, \theta] \propto \exp[\theta \cdot \phi(\text{input}, \text{label})]$$

where $\phi : \text{Inputs} \times \text{Labels} \rightarrow \mathbf{R}^d$ is a feature map. This is reminiscent of multinomial logistic regression

$$\begin{aligned} \text{score} &= \theta \cdot \phi(\text{input}, \text{label}) && \text{log-linear} \\ \text{score} &= \theta_{\text{label}} \cdot \phi(\text{input}) && \text{logistic.} \end{aligned}$$

7. A maximum entropy Markov model (MEMM) is of the form

$$\begin{aligned} P(\text{tag}_1, \dots, \text{tag}_m | \text{word}_1, \dots, \text{word}_m) \\ &= \prod_i P(\text{tag}_i | \text{tag}_1, \dots, \text{tag}_{i-1}, \text{word}_1, \dots, \text{word}_m) \\ &= \prod_i P(\text{tag}_i | \text{tag}_{i-1}, \text{word}_1, \dots, \text{word}_m) \text{ (Markov)} \end{aligned}$$

and each factor is modeled with a log-linear model

$$\begin{aligned} P[\text{tag}_i | \text{tag}_{i-1}, \text{sentence}] &\propto \\ &\exp[\theta \cdot \phi(\text{sentence}, i, \text{tag}_{i-1}, \text{tag}_i)]. \end{aligned}$$

The most likely tag sequence can be estimated with the Viterbi algorithm. MEMMs differ from HMMs in the use of features (they are difficult to add to a HMM).

A **conditional random field (CRF)** is a big log-linear model, in which the feature vector $\Phi(\text{sentence}, \text{sequence of tags})$ is of the form

$$\begin{aligned} \Phi(\text{sentence}; \text{tag}_1, \dots, \text{tag}_m) &= \\ &\sum \phi(\text{sentence}, i, \text{tag}_{i-1}, \text{tag}_i). \end{aligned}$$

It can be used with the Viterbi algorithm and fitted by gradient descent (the big sum in the normalizing constant can be dealt with with the forward-backward algorithm).

One can also consider *trigram MEMM*, modeling $P[\text{tag}_i | \text{tag}_{i-1}, \text{tag}_{i-2}, \text{sentence}]$, and generalizing trigram HMM $P[\text{tag}_i | \text{tag}_{i-1}, \text{tag}_{i-2}]$. Features can include

$$\begin{aligned} x_i &= W \wedge \text{tag}_i = T \\ \text{suffix}(x_i, 3) &= \text{ABC} \wedge \text{tag}_i = T \\ \text{prefix}(x_i, 3) &= \text{ABC} \wedge \text{tag}_i = T \\ \text{tag}_i &= T_1 \wedge \text{tag}_{i-1} = T_2 \wedge \text{tag}_{i-2} = T_3 \\ \text{tag}_i &= T_1 \wedge \text{tag}_{i-1} = T_2 \\ \text{tag}_i &= T \\ x_{i-1} &= W \wedge \text{tag}_i = T \\ x_{i-2} &= W \wedge \text{tag}_i = T \\ x_{i+1} &= W \wedge \text{tag}_i = T \\ x_{i+2} &= W \wedge \text{tag}_i = T \\ x_i &\text{ contains } [: \text{digit}:] \wedge \text{tag}_i = T \\ x_i &\text{ contains } [: \text{upper}:] \wedge \text{tag}_i = T \\ x_i &\text{ contains } [-] \wedge \text{tag}_i = T. \end{aligned}$$

8. The naive Bayes model assumes $X_i \perp\!\!\!\perp X_j \mid Y$:

$$\begin{aligned} P(Y = y, X_1 = x_1, \dots, X_n = x_n) &= \\ P(Y = y) \prod_i P[X_i = x_i | Y = y] \end{aligned}$$

i.e.,

$$p(y, x_1, \dots, x_n) = q(y) \prod_i q_i(x_i | y).$$

The maximum likelihood estimator turns out to be the sample frequencies. If the labels Y are not observed, it is a clustering problem (not unlike k -means, but with discrete variables), which can be tackled with the EM algorithm:

- E-step: estimate the cluster membership probabilities;
- M-step: estimate the parameters.

9. The Viterbi algorithm uses dynamic programming to compute the most likely sequence of hidden states in a HMM, with $T_{i,j}$ = probability of the most likely path of hidden states emitting $o_{i:j}$ and ending in state x_i .

The **forward-backward algorithm** computes

$$\sum_{s:s_j=a} \psi(s) \text{ and } \sum_{\substack{s:s_j=a \\ s_{j+1}=b}} \psi(s)$$

where $\psi(s) = \prod \psi(s_{j-1}, s_j, j)$,

$$\begin{aligned} \psi(s_{j-1}, s_j, j) &= t(s_j | s_{j-1}) e(x_j | s_j) && \text{(HMM)} \\ \psi(s_{j-1}, s_j, j) &\propto \exp[\theta \cdot \phi(x_{1:m}, s_{j-1}, s_j, j)] && \text{(CRF)}. \end{aligned}$$

Those quantities can be used to compute marginal probabilities.

It can be generalized to sequences of parse trees: the *inside-outside algorithm* allows the EM estimation of PCFGs.

Workflow control-flow patterns

A revised view

R. Russell et al. (2006)

Dependencies between tasks are often represented by a directed acyclic graph (e.g., in a **Makefile**), but more complicated dependencies are sometimes needed (sequence, and-split, and-join, or-split, or-join, first-to-complete-or-join, m-out-of-n-or-join, cancellations, interleaving, etc.); they can be modeled with *coloured Petri nets* (CPN).

Temporal evolution of financial-market correlations

D.J. Fenn et al. (2011)

The inverse participation ration of the k th principal component \mathbf{w}_k is $\text{IPR} = \sum_i w_{ik}^4$; the participation ratio, $1/\text{IPR}$, is the effective number of assets contributing to the k th component.

The evolution of boosting algorithms
A. Mayr et al. (2014)

Bagging fits a model on random subsets of the data and averages their forecasts. *Boosting* uses a similar idea, with weighted subsets (giving more weight to currently misclassified observations) and a weighted average of the forecasts. *Statistical boosting* progressively builds an (interpretable) statistical model, instead of averaging forecasts. For instance, if the weak learners are 1-variable GAMs, each boosting iteration adds one variable to the model, in a way similar to the lasso (it provides variable selection and a regularized path). While gradient boosting finds the best (incremental) candidate $h_i(x_i, \beta)$ among $h_1(x_1, \beta), \dots, h_k(x_k, \beta)$ and moves the model slightly in its direction, by adding $\varepsilon h_i(x_i, \beta)$, *likelihood-based boosting* looks at penalized $h_i(x_i, \beta)$, each maximizing $\log \text{lik}(\beta) - \text{penalty}(\beta)$, and adds the best one – thanks to the penalty, there is no need to multiply by ε .

R implementations include `mboost`, `gbm` (gradient boosting); `GAMBoost`, `CoxBoost` (likelihood-based boosting).

***Teaching logic
using a state-of-the-art proof assistant***
C. Kaliszyk et al. (2007)

Coq is used to teach logic, to avoid the almost-but-not-completely-right proofs many students generate, after defining Coq tactics that exactly match the rules of logic as they are taught.

***jHoles: a tool for understanding
biological complex networks
via clique weight rank persistent homology***
J. Binchi et al. (2014)

Look at the persistent homology of the filtered simplicial complex associated to the truncated graphs of a weighted graph. (jHoles converts the weighted graph into a filtered simplicial complex and gives it to JavaPlex to compute the persistent homology.)

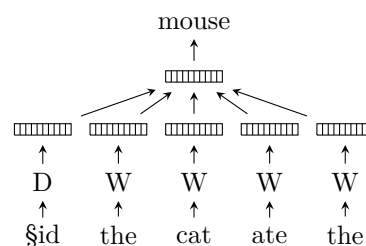
Infinite-dimensional word embeddings
E.T. Nalisnick and S. Ravi (2016)

Many statistical or machine learning models have a latent state, with values in a finite-dimensional vector space. One can also use an infinite-dimensional one, e.g., ℓ^2 , by adding a per-dimension penalty.

***Distributed representations
of sentences and documents***
Q. Le and T. Mikolov (2014)

Word2vec can be generalized to paragraphs: instead of predicting the next word from the previous words, keep the word weights w fixed and add the paragraph id as a predictor: the paragraph weights give a vector

representation of each paragraph.



***Topological pattern recognition
for point cloud data***
G. Carlsson (2013)

Gentle introduction to topological data analysis (TDA), with more examples than the other review articles.

To gain insight on the structure of a (contractible) metric space, transform it in some way, e.g., by removing a point, adding a point (1-point compactification, for separated, locally compact, non-compact spaces), removing singular points, removing the “center” (*i.e.*, keeping the “end points”) to identify appendages.

Functional persistence homology is the persistence homology of $f^{-1}(\alpha)$ or $f^{-1}([\alpha, +\infty])$, for some map $f : X \rightarrow \mathbf{R}$ (e.g., the centrality).

Applications include:

- Chemistry: the functional persistence homology barcodes measure how close two molecules (clouds of point/atoms) are;
- Genetics: non-trivial homology of a set of (viral) genetic sequences (for the Hamming distance) reveals horizontal gene transfer;
- Time series: the homology of the set of (centered, normalized) fragments $(x_t, x_{t+1}, \dots, x_{t+k})$ can help identify periodic time series;
- Cosmology: persistence homology (in particular the Euler characteristic) can be used to compare the structure of the universe (galaxies form clusters, filaments, walls) with that from a Gaussian random field.

Introduction to the R package TDA
B.T. Fasy et al.

***Homology and cohomology computation
in finite element modeling***
M. Pellikka et al.

`gmsh` can compute homology and cohomology – for some PDE problems, the boundary only determines the solution within a given cohomology class.

***PREMiuM: an R package for profile regression
mixture models using Dirichlet processes***
S. Liverani et al. (JSS, 2015)

Profile regression is a mixed model in which the groups come from a Dirichlet process mixture clustering.

***GPfit: an R package
for fitting a Gaussian process model
to deterministic simulator outputs***
B. MacDonald et al. (JSS, 2015)

When fitting a Gaussian process on non-noisy data, with a Gaussian correlation function $R_{ij} = \text{Cor}(y_i, y_j) = \prod_k \exp -\theta_k |x_{ik} - x_{jk}|^2$, $\theta_k > 0$, the correlation matrix can be ill-conditioned (some of the points are too close), and the likelihood has local extrema very close to zero. One can replace R with $R + \delta I$, with the smallest nugget δ that makes the matrix well-conditioned, and reparametrize θ as $\theta = \exp \phi$.

Check the R packages `GPfit` (no noise), `tgp` (noise) or `mlegp` (noise, numerically unstable).

Also mentions `lhs::maximinLHS` (random but well-dispersed points).

***Introducing multivalor:
a multivariable emulator***
R.K.S. Hankin

Gaussian processes in higher dimensions.

***Newton's versus Halley's method:
a dynamical systems approach***
G.E. Toberts and J. Horgan-Kobelski (2003)

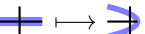
To solve $f(x) = 0$, Newton's method approximates f with an affine function, Halley's method with a hyperbola.

Pricing composable contracts on the GP-GPU
J. Ahnfelt-Rønne and M.F. Werk (2011)

SPL (stochastic process language) is a Haskell DSL to describe and (efficiently) price derivatives.

***Fractional-parabolic deformations
with sinh-acceleration***
S. Levendorskiĭ

Some integrals can be computed using:

- A fractional-parabolic deformation, *i.e.*, by replacing an integral over \mathbf{R} with an integral over $\sigma \exp[\alpha \log(1 + i\mathbf{R})]$: 
- Sinh acceleration:

$$\begin{aligned} \int_{\mathbf{R}} f(x) dx &= \int_{\mathbf{R}} f(\sinh y) \cosh y dy \\ &\approx \sum_{|n| \leq N} f(\sinh n) \cosh n \end{aligned}$$

***Market timing and return prediction
under model instability***
M.H. Pesaran and A. Timmermann (2002)

To detect structural breaks, apply CUSUM tests to observations reversed in time.

***On the equivalence between quadrature rules
and random features***
F. Bach (2015)

The quadrature problem can be formulated as the problem of approximating an element $i : f \mapsto \int_D f$ of a Hilbert space as a linear combination of well-chosen elements $\text{ev}_x : f \mapsto f(x)$.

It is similar to the kernel approximation problem: finding a low-dimensional embedding ϕ such that $k(x, y) \approx \langle \phi(x), \phi(y) \rangle$.

The CMA evolution strategy: a tutorial
N. Hansen (2011)

CMA-ES is a population-based optimization algorithms that models the current population as a Gaussian, samples from this model, keeps the best candidates and iterates.

Gaussian processes for machine learning
C.E. Rasmussen and C.K.I. Williams (2006)

A Gaussian process is an infinite family of random variables $(X_t)_{t \in \mathbf{R}}$; the mean and covariance functions

$$m(t) = E[X_t], \quad k(s, t) = \text{Cov}(X_s, X_t)$$

suffice to define it. The joint distribution of X_{t_1}, \dots, X_{t_n} is Gaussian, and the conditional distribution $X_{s_1}, \dots, X_{s_m} | X_{t_1}, \dots, X_{t_n}$, obtained in the usual way (Shur complement) is the posterior distribution.

To train a Gaussian process, one can use a hierarchical prior, *i.e.*, posit a parametrization of the mean and covariance functions, *e.g.*,

$$\begin{aligned} m(t) &= at^2 + bt + c \\ k(s, t) &= \sigma_1^2 \exp -\frac{(s-t)^2}{\ell^2} + \sigma_2^2 \delta_{ij} \end{aligned}$$

(the $\sigma_2^2 \delta_{ij}$ term accounts for noisy observations) and select the hyperparameters via log-marginal likelihood.

***Time series analysis using Gaussian processes
in Python and the search for Earth 2.0***
D. Foreman-Mackey (PyData, 2014)

Application of Gaussian processes to detect transits of extrasolar planets.

***Portfolio optimization
for VAR, CVaR, omega and utility
with general return distributions***
W.T. Shaw

Biased random portfolios.

Algorithms for hyper-parameter optimization
J. Bergstra et al. (2011)

A **Parzen estimator** of a conditional probability distribution $p(x|y)$ is an estimator of the form

$$p(x|y) = \begin{cases} \ell(x) & \text{if } y < y^* \\ g(x) & \text{if } y \geq y^* \end{cases}$$

where ℓ and g are kernel estimators (possibly with a Gaussian, etc. prior) and the threshold y^* is some quantile of the sample data $p(y < y^*) = \gamma$.

The expected improvement (x = parameter, y = loss) is (an increasing function of) $\ell(x)/g(x)$: to get a new candidate solution, one can sample many points from ℓ and take that with the highest $\ell(x)/g(x)$.

The parameter space of hyperparameter optimization problems often has a tree structure: the Parzen estimator can be generalized to a *tree-structured Parzen estimator* (TPE).

Sequential search can be parallelized with the **constant liar** approach: once a candidate point has been chosen, provisionally set its loss to the average loss.

Check the **hyperopt** Python package.

**Sequential model-based optimization
for general algorithm configuration**
F. Hutter et al. (2011)

Gaussian processes (GP) can be generalized to mixed continuous-categorical variables:

$$k(x, y) = \exp \sum_j -\lambda_j d(x_j, y_j)^2,$$

where d is the Euclidean or Hamming distance (note that the coordinates are independent).

Sequential model-based optimization (SMBO, aka Bayesian optimization) can use *random forests* (of regression trees) instead of GPs.

Depending on the problem, you may want to transform the objective function (e.g., replacing the algorithm running time with its logarithm).

Instead of estimating the performance of an algorithm as a sample mean

$$\frac{\text{Mean Performance}(i, \theta)}{i \in \text{Instances}}$$

on a small set of instances, one can learn the mapping (instance, θ) \mapsto performance, using instance features, and use it to estimate $E[\text{Performance}(\cdot, \theta)]$. (You also need a distribution on instance features. SATzilla uses a similar idea to choose the most promising algorithm (DPPL, LP, etc.) for a given instance.)

To maximize the expected improvement (EI), and produce a diverse set of configurations with high EI, use multi-start local search.

**Preliminary evaluation
of hyperopt algorithms on HPOLib**
J. Bergstra et al. (2014)

Testing the hyperopt algorithms on the HPOLib benchmarking suite suggests that vanilla Gaussian processes (with RBF, rather than Matérn kernels) are good enough.

Making a science of model search
J. Bergstra et al. (2013)

TPE is slightly better than random search.

**Time-bounded
sequential parameter optimization**
F. Hutter et al. (2010)

A few improvements on SPO:

- Instead of fitting a *noise-free* Gaussian process to smoothed data (if there are several, contradictory measurements for the same parameter values, you need to somehow reconcile/smooth them), include noise in the model;
- (When minimizing running time) stop the computations early if they take too long: a *censored* running time is still informative;
- Use an approximate Gaussian process, the *projected process* (PP) approximation: instead of considering (inverting) the whole $n \times n$ kernel matrix, select (without replacement) p instances and make do with the $n \times p$ and $p \times p$ submatrices.

**SMAC:
sequential model-based algorithm configuration**

Random-forest-based Bayesian optimization, in Java.

**BayesOpt: a Bayesian optimization library
for nonlinear optimization,
experimental design and bandits**
R. Martinez-Cantin (2014)

Gaussian process Bayesian optimization, in C++.

**Efficient benchmarking
of hyperparameter optimizers via surrogates**
K. Eggenberger et al. (2015)

Comparison of three Bayesian optimization methods:

- Gaussian processes (GP, as implemented in Spearmint);
- Random forests (RF, as implemented in SMAC);
- Tree-structured Parzen density estimators (TPE, as implemented in hyperopt)

on realistic benchmarks, surrogates of real-world problems (real-world problems usually require time, software licenses, and/or specialized hardware, and are therefore hard to reproduce). On low-dimensional continuous problems, GP perform best; on more general problems, RF and TPE perform best.

Optimization: algorithms and applications
R.K. Arora (2015)

Relatively exhaustive list of all optimization-related topics, not too deep, but with examples, exercises and Matlab (?) code.

Tree-structured Gaussian process approximation
T. Bui and R. Turner

Gaussian process estimation can be sped up using a (smaller) pseudo-dataset of (noiseless) pseudo-observations; they can be arranged in a tree (à la Barnes-Hut) and used as a graphical model.

F-Race and iterated F-Race: an overview
M. Birattari et al. (2009)

To minimize an expensive function, e.g., some expected cost, estimated by Monte Carlo simulations: take a set of candidates (e.g., a grid); evaluate them in parallel; regularly test if the (more and more precise) estimated values are significantly different (e.g., with a non-parametric ANOVA test); if so, discard the worst candidate; iterate until only one candidate remains.

This can be combined with CMA-ES: when the number of candidates has been sufficiently reduced, replace them with new, nearby candidates.

ParamILS: an automatic algorithm configuration framework
F. Hutter et al.

Iterated local search (ILS) builds a sequence of local minima: start with a point; use local search to find a nearly local minimum; add some noise to escape its basin of attraction; iterate, keeping the new minimum if it is better (other acceptance criteria are possible).

If the objective function is estimated from a Monte Carlo simulation, there is no need to always use the same number of iterations.

If the objective function is the running time of some algorithm, censored values (“more than x ”, i.e., we stopped after x seconds) are sometimes good enough.

GGA: a gender-based genetic algorithm for the automatic configuration of algorithms
K. Tierney

Gender-based genetic algorithms keep two subpopulations (genders), using different fitness functions. For instance, if the standard fitness is time-consuming to compute, one could use it for one gender and use the other as a “variety store”.

R in Finance 2016

The `ForecastCombinations` package provides various ways of combining predictions:

- Equal-weighted average;

- Precision-weighted average;
- Regression;
- Regression with L^1 loss;
- Constrained regression: $\sum \beta_i = 1$, $\beta_i \geq 0$;
- Average of the best 10% subset regressions;
- BIC-weighted average of all subset regressions.

Stan, a C++-like language to describe Bayesian models, in the spirit of BUGS, is not limited to hierarchical models, but can also be used to model time series: stochastic volatility, mixtures, hidden Markov models, hierarchical stochastic volatility.

Deep learning libraries are no longer limited to Python and Lua: MxNet (a C++ library) can be used from R.

There is now an R package for automatic differentiation (AD): `madness`.

Several talks dealt with portfolio optimization:

- Multi-objective portfolio construction, with NSGA2 (`mco::nsga2`; risk parity, `cccp::rp`, was also mentioned)
- The ROML package provides portfolio optimization functions, a bit like a limited version of disciplined convex optimization, with predefined functions for many objectives. [R finally has a package for disciplined convex optimization: `cvxr`.]
- Portfolio optimization with PortfolioAnalytics and random portfolios

Most talks were about finance:

- Do not use the ADF test to identify trading opportunities: it highlights mean reversion (Ornstein-Uhlenbeck process), while you want oscillations. Try a model of the form

$$x_{t+1} = \beta_0 + \beta_1 \times \text{level}_t + \beta_2 \times \text{slope}_t + \text{noise}_{t+1};$$

filter for $\beta_1 < 0$, $\beta_2 > 0$; keep the best fits. One could also try a Hilbert transform.

- SVM for stock selection (on the S&P100): comparison of 15 different kernels [one could also combine them – this is called multiple kernel learning (MKL)].
- One can measure the connectedness of the bank network by fitting a VAR model (use an adaptive lasso penalty to deal with the large number of banks, and an OHLC-based volatility estimator) and computing the H -step ahead contribution of bank i to bank j .
- Leveraged ETFs give less return than you may think:

$$f(x) = \log_{1p}(k \times \exp_{1p}(x)) \leq k \times x.$$

This becomes more visible when volatility is high, and with compounding:

$$\sum f(x_t) \leq k \sum x_t.$$

- Building an index from GoogleTrends and a list of search terms

A few talks covered data management issues, e.g., `h5`, to read and write hdf5 files (a portable file format to store arrays of numbers, data.frames, etc.), `feather` (to store columnar data and read it from R or Python)

or `ff` to manipulate datasets larger than memory. R can be used in other environments, such as Postgres or Hadoop.

A few talks focused on efficient or extensible computations.

- The **Arborist** package for random forests also provides quantile regression (i.e., the leaves do not only contain the mean, but quantiles) and monotonicity constraints (reject proposals that breach the constraints with probability p – not necessarily 1).
- The **roll** package provides more rolling functions (`mean`, `var`, `sd`, `cor`, `cov`, `lm`, `eigen`, `pcr`, `vif`); the computations use **Rcpp** and are parallelized.
- The **pirls** package re-implementats `glmnet`; it is slower but allows arbitrary link functions.

Many talks showcased Shiny, interactive (Javascript) plots and the use of R to teach finance or options.

There were a few talks about options and stochastic processes:

- The illiquidity of a market can be estimated by comparing the price of an ETF and the NAV of the underlying portfolio; this can be interpreted as a bid-ask spread by noticing that bid and ask prices are American option prices.
- Computation of the returns of the dual moving average (MACD) strategy, under a Brownian motion assumption
- Convertible bonds, divergence swaps, stochastic local volatility model, etc.

A few talks were only remotely related to finance. For instance, the *rearrangement problem* is the problem of finding the intra-column rearrangement of a matrix that minimizes row sum variability (variance, maximum, minus minimum). Here are a few heuristics:

- Pick a column at random and re-order it so that it be in the order opposite to the sum of the columns;
- Idem with a subset of columns versus the others;
- This subset of columns can be chosen so that its sum has as similar a variance as the other columns.

Why should I trust you?

Explaining the predictions of any classifier

M.T. Ribeiro et al. (2015)

To interpret the forecasts of a complex (black box) model, approximate it, locally (as in local regression) with simpler models: sparse linear model (e.g., for a small number of binary features), decision trees, falling rule lists, etc. The explanation is *local*: each observation gets a different explanation. To explain the whole model, provide a diverse, representative set of observations and explanations; it is a (weighted) *set cover problem*: given an instance \times feature binary (or weight) matrix, select a small number of rows that cover as many columns as possible (a greedy algorithm is good enough).

Falling rule lists

F. Wang and C. Rudin (2014)

A *falling rule list* is a set of rules of the form “if patient satisfies condition 1, then risk = $x\%$ else...”, where the risk is decreasing. From a set of candidate conditions B (frequent item set mining: FP-growth, Eclat, Apriori), one can build a Bayesian model (greedily selected decision rules do not perform well).

B : set of candidate rules

λ : expected number of rules

w_j : prior probability of $j \in B$

$\Gamma(\alpha_j, \beta_j)$: prior for the risk of rule j

$\Gamma(\alpha_\infty, \beta_\infty)$: prior for the risk of the default rule

$L \sim \text{Pois}(\lambda)$ number of rules

$c_1 \dots, c_L$: drawn from B , without replacement, with probabilities proportional to w_j

$\gamma_j \sim \Gamma(\alpha_j, \beta_j) \mathbf{1}_{\geq 1}$

$\gamma_\infty \sim \Gamma(\alpha_\infty, \beta_\infty)$

$\text{risk}_j = \sum_{i \geq j} \log \gamma_i$

Interpretable classifiers using rules and Bayesian analysis: building a better stroke prediction model

B. Letham et al. (2015)

Bayesian rule lists generalize falling rule lists to multinomial outcomes.

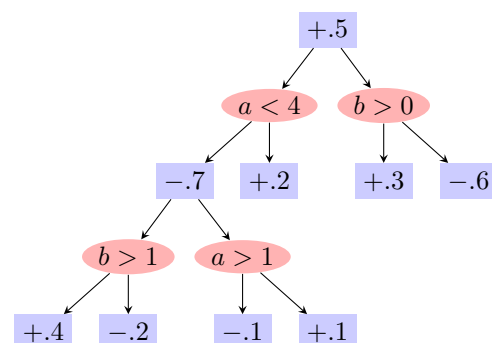
Scalable Bayesian rule lists

H. Yang et al. (2016)

The alternating decision tree learning algorithm

Y. Freund and L. Mason (1999)

An **alternating decision tree** alternates between prediction nodes and splitting nodes; the value of a leaf is the sum of the predictions on its path; the value of a sample is the sum of the values of the paths leading to it.



They can be learnt by boosting; their performance is similar to boosted decision trees (C5.0), but they are easier to interpret.

**Optimizing the induction
of alternating decision trees**
B. Pfahringer et al.

Alternating decision trees are not scalable: heuristics are needed.

Ensemble samples with affine invariance
J. Goodman and J. Weare (2010)

The affine-invariant Nelder-Mead optimization algorithm motivates an affine-invariant *ensemble* of MCMC samplers, using a “stretch move” (adjust the acceptance probability accordingly)

$$x_k^{\text{candidate}} = x_j + Z(x_k - x_j), \quad j \neq k \text{ random}$$

$$Z \sim p, \quad p(z) \propto \frac{1}{\sqrt{z}} \mathbf{1}_{[1/a, a]}, \quad a = 2$$

Similarly, CMAES suggests a “walk move”,

$$x_k^{\text{candidate}} \sim N(\mu_S, \Sigma_S)$$

where S is a random subset of particles, $|S| \geq 2$, $k \notin S$, and μ_S and Σ_S are their mean and variance.

This is implemented in the `emcee` Python package.

Integration in finite terms
M. Rosenlicht (1972)

Elementary proof of Liouville’s 1834 theorem, which gives a criterion for an indefinite integral to be expressible in closed form. For instance, $\int e^{-x^2} dx$ has no closed form

An invitation to integration in finite terms
E.A. Marchisotto et G.A. Zakeri (1994)

More elementary presentation of Liouville’s theorem.

**Smooth numbers:
computational number theory and beyond**
A. Granville (2008)

Smooth numbers are numbers whose prime factors are small. A number is *y-smooth* if its prime factors are at most y . Let $\psi(x, y)$ be the number of y -smooth integers up to x . Then

$$\frac{\psi(x, x^{1/u})}{x} \xrightarrow{x \rightarrow \infty} \rho(u) > 0$$

where

$$\begin{aligned} \rho(u) &= 1 & u &\in [0, 1] \\ \rho(u) &= 1 - \log u & 1 &\leq u \leq 2 \\ \rho(u) &= \frac{1}{u} \int_{u-1}^u \rho(t) dt & u &> 1. \end{aligned}$$

They play a role in cryptography and computational number theory.

**Generalizing dynamic time warping
to the multidimensional case
requires an adaptive approach**
M. Shokoohi-Yekta et al.

Dependent and independent multidimensional time warp (DTW) give different results. Which is best may be problem-, class- or exemplar-specific, and can be learned.

**Clustering time series
using unsupervised shapelets**
J. Zakaria et al.

(After normalizing the time series) apply clustering algorithms to the matrix of distances between the time series and “shapelets” (shorter time series); a small number of discriminative shapelets suffices.

Foster-Hart optimal portfolios
A. Anand et al.

For a bounded random variable X such that $E[X] > 0$ and $P[X < 0] > 0$, the Foster-Hart risk is the positive number r such that $E \log(1 + X/r) = 0$. It is not coherent, not convex, not time-consistent, not guaranteed to exist.

**Option-implied equity premium predictions
via entropic tilting**
K. Metaxoglou et al. (2016)

Entropy tilting transforms a probability distribution π to change some of its moments g .

$$\begin{aligned} \text{Find} & \quad \pi^* \\ \text{To minimize} & \quad \text{KL}(\pi^* || \pi) = \int \pi^* \log \frac{\pi^*}{\pi} \\ \text{Such that} & \quad \int g(x) \pi^*(x) dx = \bar{g} \end{aligned}$$

Using semantic fingerprinting in finance
F. Ibriyamova et al.

Rather than cosine similarity between bags of words, use vector embeddings (here, Numenta’s “semantic fingerprint”) to identify similar companies.

Pure quintile portfolios
D. Liu

Let r be the vector of stock returns (at a given date), X the matrix of (normalized) factor exposures (stock \times factor); the *factor returns* are $(X'X)^{-1}X'r$; the *factor mimicking portfolios* are the rows of $(X'X)^{-1}X'$. The *factor quintile portfolios* are the minimum variance portfolios subject to the constraints

- (i) $n/5$ stocks, long-only;
- (ii) The exposure to the factor of interest is the same as that of the quintile portfolio;
- (iii) The exposure to the other factors is zero.

***Modeling and forecasting
(un)reliable realized covariances
for more reliable financial decisions***
T. Bollerslev et al. (2016)

The realized covariance is the sum of a (time-dependent but persistent) unobserved variance, and a noise term (market microstructure, Epps effect, non-synchronous observations). Estimating the distribution of this noise term leads to a better (exponentially-weighted) covariance estimator.

***Equivalence
of robust VaR and CVaR optimization***
S. Lotfi and S.A. Zenios (2016)

Robust optimization

$$\underset{\alpha}{\text{Minimize}} \underset{\beta \in B}{\text{Max Loss}}(\alpha, \beta)$$

optimizes the worst case situation, when the values of some parameters are constrained to remain in some box or ellipsoid. Robust CVaR and robust VaR optimization turn out to be equivalent.

An introduction to ROC analysis
T. Fawcett (2005)

Precision-recall curves are affected by class skew; ROC curves are not.

Given a set of classifiers, look at the convex hull of their ROC curves; given two classifiers A and B , the random mixtures $pA + (1-p)B$ form the $[AB]$ segment.

The AUC can be interpreted as the probability, given two observations from the two classes, of correctly distinguishing them.

To estimate the “precision” of a ROC curve, use bootstrapped curves and look at the variation of TP given FP, or (FP, TP) given the score.

For multiclass problems, use $\sum p(c_i) \text{AUC}(c_i)$ or

$$\frac{2}{|C|(|C| - 1)} \sum_{i < j} \text{AUC}(c_i, c_j)$$

***Non-linear dimensionality reduction:
Riemannian metric estimation
and the problem of geometric recovery***
D. Perrault-Joncas and M. Meilă

By inverting the Laplace-Beltrami operator, one can add a Riemannian metric to manifold algorithms, showing how distances, angles and areas are distorted. [One could also try to minimize those distortions.]

***Bayesian model choice and information
criteria in sparse generalized linear models***
R. Foygel and M. Drton

The **extended BIC** uses a prior favouring sparser models.

$$P(J) \propto \binom{p}{|J|}^{-\gamma} |J| \leq q \quad J \subset \llbracket 1, p \rrbracket$$

$$\text{BIC}_\gamma(J) = -2 \log \text{lik}(\hat{\theta}_J) + |J| \log n + 2\gamma |J| \log p$$

***Extended Bayesian information criteria
for model selection with large model spaces***
J. Chen and Z. Chen

Original paper on the eBIC.

***A new method for constructing
networks from binary data***
C.D. van Borkulo et al. (2014)

The structure of a Gaussian Markov random field (MRF, *i.e.*, an undirected graphical model) can be estimated by a graphical lasso (estimate the inverse of the covariance matrix, whose nonzero entries correspond to conditional independencies) and approximated by lasso regressions, one for each variable. This approximation can be generalized to binary MRF (Ising model) and lasso logistic regressions.

***High-dimensional graphical model selection
using ℓ_1 -regularized logistic regression***
M.J. Wainwright et al.

***Confidence bounds of recurrence-based
complexity measures***
S. Schinkel et al. (2009)

Recurrence plots highlight periodic or chaotic behaviour in time series.

$$R_{ij} = \begin{cases} 1 & \text{if } \|x_i - x_j\| \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

Recurrence quantification analysis (RQA) turns those plots into numbers (features):

- Proportion of dark points (recurrence rate);
- Proportion of points in diagonal segments of length at least ℓ_{\min} (determinism);
- Length of the longest diagonal segment;
- Average length of the diagonal segments;
- Similar quantities for vertical segments (laminarity, trapping time).

To compute confidence intervals on those quantities, resample, with replacement, the diagonal (resp. vertical) segments (including those of length one).

***The PP-TSVD algorithm
for image restoration problems***
P.C. Hansen et al.

Besides the penalized

$$\begin{array}{ll} \text{Find} & z \\ \text{To minimize} & \|Az - b\|_2^2 + \lambda \|z\|_2 \end{array}$$

and constrained problems

$$\begin{array}{ll}\text{Find} & z \\ \text{To minimize} & \|Az - b\|_2^2 \\ \text{Such that} & \|z\|_2 \leq c\end{array}$$

one can consider the truncated one

$$\begin{array}{ll}\text{Find} & z \\ \text{To minimize} & \|z\|_2 \\ \text{Such that} & \|Az - b\|_2^2 \leq c.\end{array}$$

As usual, one can replace $\|z\|_2$ with $\|z\|_1$, $\|\nabla^2 z\|_2$, $\|\nabla z\|_2$, $\|\nabla z\|_1$, etc.

***The evolution of popular music:
USA 1960-2010***
M. Mauch et al. (2015)

To study a musical corpus:

- Extract timbre features (12 Mel-frequency cepstral coefficients (MFCC), $\Delta\text{MFCC} = \text{MFCC}_0(t) - \text{MFCC}_0(t-1)$, zero-crossing count) using some Vamp plugin; extract the 12 chroma features from NNLS Chroma, for each frame;
- Apply random sampling (20 frames), PCA, model-based clustering (Gaussian mixture, 35 clusters) to build the timbre lexicon;
- Use the common chord transitions as a harmony lexicon;
- Extract topics with latent Dirichlet allocation (LDA, with the `topicmodels` package).

***An end-to-end neural network
for polyphonic music transcription***
S. Sigtia et al.

Automatic music transcription (AMT) can benefit from the same tools as speech recognition: language models, hidden Markov models, recurrent neural nets, convolutional nets.

Exploiting spillovers to forecast crashes
F. Gresnigt et al. (2015)

Add cross-excitations to Hawkes processes, to model contagion.

fastFM: a library for factorization machines
I. Bayer

Factorization machines

$$y = \alpha + \sum \beta_i x_i + \sum_{i < j} \langle \gamma_i, \gamma_j \rangle x_i x_j$$

are not limited to recommendation systems.

Implementations include `fastFM`, `libFM`, `libffm`, `GraphLab`.

A neural algorithm of artistic style
L.A. Gatys et al.

Compute the correlations between different features in different layers of the CNN and try to match them to those of a “target style” image.

***Compressing neural networks
with the hashing trick***
W. Chen et al.

Use Vowpal Wabbit’s hashing trick for weight sharing in (large) neural nets.

Generalized communities in networks
M.E.J. Newman and T.P. Peixoto

In the stochastic block model, each node belongs to a community, and the link probability only depends on the endpoints’ communities. It can be generalized: each node is assigned a random number, and the link probability only depends on those numbers.

$$\begin{aligned}x_u &\sim U(0, 1) \\ p_{uv} &= \omega(x_u, x_v)\end{aligned}$$

To account for non-Poisson degree distributions, consider instead

$$p_{uv} = \frac{d_u d_v}{2 \times \# \text{edges}} \omega(x_u, x_v).$$

The model can be efficiently estimated (EM with belief propagation, expansion of ω in the basis of Bernstein polynomials – they are nonnegative over $[0, 1]$) and uncovers both community structures and spatial structures.

***Gradient-free Hamiltonian Monte Carlo
with efficient kernel exponential families***
H. Strathmann et al.

Metropolis-Hastings MCMC samples from a probability distribution π , known only up to a multiplicative factor.

$$\begin{aligned}x^* &\sim Q(\cdot|x_t) \text{ proposal distribution} \\ x_{t+1} &\leftarrow x^* \text{ with probability } \min \left\{ 1, \frac{\pi(x^*)}{\pi(x_t)} \frac{Q(x_t|x^*)}{Q(x^*|x_t)} \right\} \\ x_{t+1} &\leftarrow x_t \text{ otherwise}\end{aligned}$$

The proposal distribution is usually Gaussian, with covariance

$$\Sigma_t \propto I = E[(\nabla \log \pi)^2] = -E[\nabla^2 \log \pi].$$

If $\nabla \log \pi$ is intractable, the information matrix can be estimated from the history of the Markov chain, with a kernel embedding if there are nonlinear effects (kernel adaptive MH, KAMH). These ideas can be generalized to Hamiltonian Monte Carlo (HMC).

***A Riemannian framework
for tensor computing***
X. Pennec et al. (2005)

Many algorithms from statistics or numerical analysis can be generalized to the manifold of positive definite matrices S_+ (“tensors”), endowed with the Riemannian structure (Fisher information metric) induced by $\exp : \text{Sym} = T_{\text{Id}} S_+ \rightarrow S_+$, invariant by

$$\begin{cases} \text{GL} \times S_+ \longrightarrow S_+ \\ (X, A) \longmapsto XAX'. \end{cases}$$

For instance:

- Mean: $\sum_i \log_{\bar{x}} x_i = 0$;
- Gradient descent: $x_{t+1} \leftarrow \exp_{x_t}[-\varepsilon \nabla C(x_t)]$;
- Linear interpolation (geodesic):

$$x(t) = \exp_{x_1}[t \overrightarrow{x_1 x_2}] = \exp_{x_1}[t \log_{x_1} x_2]$$

where

$$\begin{aligned} \exp_x(y) &= x^{1/2} \exp(x^{-1/2} y x^{-1/2}) x^{1/2} \\ \log_x(y) &= x^{1/2} \log(x^{-1/2} y x^{-1/2}) x^{1/2}. \end{aligned}$$

***Laplace-Beltrami eigenvalues
and topological features of eigenfunctions
for statistical shape analysis***
M. Reuter et al. (2009)

Use the spectrum (“shape DNA”) of the Laplace operator, and its eigenfunctions (level sets, critical points) as features for shape classification or comparison.

***Geometric understanding of point clouds
using Laplace-Beltrami operator***
J. Liang et al.

The Laplace operator can be approximated, from a cloud of points, using local least squares.

***On the origin of burstiness in human
behavior: the Wikipedia edits case***
Y. Gandica et al. (2016)

For inter-event times, compute the burstiness

$$B = \frac{\sigma - \mu}{\sigma + \mu} \in [-1, 1]$$

(periodic if $\sigma \ll \mu$, bursty if $\sigma \gg \mu$) and the autocorrelation.

What is a paraproduit?
A. Bényi et al. (2010)

A generalization of the operator

$$\Pi_0(f, g)(s) = \int_{-\infty}^s f' g.$$

***Surprising patterns for the call duration
distribution of mobile phone users***
P.O.S. Vaz de Melo et al.

The call duration distribution has heavier tail and head than the log-normal, but can be modeled as a log-logistic,

$$P[X \leq x] = \frac{1}{1 + e^{-z}}, \quad z = \frac{\log x - \mu}{\sigma}.$$

***Fast moment-based estimation
for hierarchical models***
P.O. Perry (2014)

Moment methods can fit large hierarchical models (e.g., a recommender system with 5 predictors, 1000 groups, 100,000 observations).

***Differential geometric least angle regression:
a differential geometric approach
to sparse generalized linear models***
L. Augugliaro and A.M. Mineo (2010)

The GLM L^1 regularization path is not piecewise linear but, on the statistical manifold of an exponential family, the least angle regression algorithm (keep the angle between the “score” of the active variables and the residual tangent vector constant) yields a sparse regularization path. This is implemented in the `dglars` package.

***Convolutional neural networks
for visual recognition***
A. Karpathy and J. Johnson (2016)

Simple classifiers such as k -NN, softmax (multinomial logistic) or multiclass SVM (all with regularization) do not perform well on image classification tasks (MNIST, Cifar-10, ImageNet), though image features (colour histogram, bag of visual words, HOG, SIFT, GIST, LBP, Texton, SSIM, etc.) help: prefer deep convolutional nets, with ReLU units.

To address the vanishing/exploding gradient problem, initialize the weights so that the activations be approximately standard Gaussian: for instance, Xavier initialization suggests `rand()/sqrt(fan-in)`, for linear (or tanh) units and `rand()/sqrt(fan-in/2)` for ReLU. Alternatively, insert *batch normalization layers* to rescale the activations – it is a differentiable operation, therefore amenable to back-propagation. The rescaling is mini-batch-specific: the jittering effect is similar to that of dropout. To use the network, after training, use the (fixed) μ and σ from the whole training set.

To debug your network:

- The initial loss should be $\log(\# \text{classes})$ without regularization, more with
- The unregularized network can overfit a small dataset
- Plot loss and test accuracy versus epoch and weight update; $|\text{weight update}|/\text{weight magnitude}$ should be around 10^{-3}

- Use random hyperparameter search; worry if the best ones are on the boundary.

Among the many optimization algorithms,

- Stochastic gradient descent is slow

$$x += -\alpha dx$$

- Momentum tends to overshoot

$$v = \mu v - \alpha dx$$

$$x += v$$

- Nesterov momentum (aka Nesterov accelerated gradient, NAG) evaluates the gradient after the momentum step
- With Adagrad, the step sizes tend to zero too fast

$$N += (dx)^2$$

$$x += -\alpha dx / \sqrt{N}$$

- RMSprop is similar to Adagrad, but uses an exponential moving average of the second moment instead of a sum;

prefer Adam, which combines momentum and RMSprop, *i.e.*, the first and second moments of the gradients.

Ensembling helps: even if you have a single model, you can simply average some of its snapshots, taken during training, or take an exponential moving average of its weights over training.

Dropout suggests to drop half the neurons at random, at train time, and shrink the activations at test time. Instead, one could drop connections (DropConnect).

Zero-padding is necessary, otherwise the image shrinks at each step, which forbids deep nets. The 1×1 filters make sense – they are actually $1 \times 1 \times \text{depth}$.

Pooling layers are often inserted after some of the convolutional layers: they downsample the image, usually with max pooling (average does not work as well).

Conv nets usually end with a fully connected layer, *e.g.*, a softmax.

Historical convnets include LeNet, AlexNet, ZFNet, VGGNet and, more recently, GoogLeNet (no FC layers), ResNet (152 layers, with skip-layer connections) – pretrained models are available in the Caffe model zoo.

Pre-trained convnets can tackle many computer vision tasks by replacing the “classification head” with another one: localization, detection, etc.

To understand what a convnet does:

- Pick a neuron, see what images excite it;
- Only the first-layer weights are interpretable: they are always Gabor-like features;
- The output of the FC7 layer (last before the classifier) gives a “code” for the image: visualize the cloud of images with t-SNE.

- Occlude a part of the image and check how the probability changes, as a function of the region occluded, to see where the object is;
- Deconv computes the gradient of a specific neuron wrt all the pixels to see what it sees; in guided backprop: ReLU units let the gradient pass if gradient, or both input and gradient are positive (instead of input only) – this gives nicer interpretations of the neurons;
- Choose a neuron and build an image that maximizes its output; regularize with an L^2 penalty or blur the image after each step to avoid accumulation of high-frequency features.

Deep dream starts with an image and tries to amplify the features in a given layer (forward pass until the target layer, set the gradient to the activations, backprop the gradient back to the image, update the image, iterate). *NeuralStyle* (deep art) looks for an image close to the contents of one image (measured by the activations of a layer) and the style of another (measured by the Gram (covariance) matrix of one or several layers).

With *recurrent networks*, do not backprop through the whole sequence, but only 25 characters at a time; to avoid exploding gradients, cut them off. The LSTM or GRU (a slight simplification) equations look complicated: changing them at random does not affect performance

CNNs do not necessarily need a lot of data: take a pre-trained net and freeze all but the last few layers – it becomes a feature extractor and you can forget that the features come from a neural net.

To reduce the number of parameters and increase expressivity,

- Replace large convolutions (5×5 , 7×7) with stacks of 3×3 convolutions;
- Apply a 1×1 convolution to halve the number of filters, then a 3×3 convolution, then a 1×1 to restore the number of filters (bottleneck convolution);
- Replace a 3×3 convolution with a 1×3 followed by a 3×1 .

To increase the training set (data augmentation), use horizontal flips, crops (train the network on crops; use crops at test time and average several crops), or ajitter the colour.

Convolutions can be expressed as matrix multiplications and sped up by linear algebra libraries (to the price of some data duplication); FFT only helps for large filters.

Double precision is not needed: single or half is enough.

Semantic segmentation is the task of labeling each pixel (“cow”, “tree”, “person”, etc.); instance segmentation also distinguishes different instances (first cow, second cow, etc.) Semantic segmentation networks usually end with an upsampling operation, which can be learnt (fractionally-strided convolution, often improperly called “deconvolution”).

For image segmentation, generate a set of proposals (regions of interest, ROI), and refine them (grey out pixels far away and predict which of the remaining pixels are in the region).

For video, detect feature points at different scales; track them using optical flow; extract features (HOG, HOF, MBH) in the corresponding stabilized coordinates. Spatio-temporal convnets use 3-dimensional convolutions, but single-frame CNNs (on just one frame – discard the others) perform well, and video does not add much.

Variational auto-encoders add a Bayesian flavour to auto-encoders: start with data, x ; the encoder generates μ_z, σ_z ; sample $z \sim N(\mu_z, \Sigma_z)$; the decoder generates μ_x, σ_x ; sample $x \sim N(\mu_x, \Sigma_x)$; the loss function combines the reconstruction loss on x and a penalty to ensure that $N(\mu_z, \Sigma_z)$ is close to the prior on z .

Generative adversarial networks train two networks together: the generator turns random noise into a candidate image; the discriminator is given an image and must tell whether it comes from the data or was generated. They work well on simple datasets (MNIST, faces); a multiscale variant also works on more complex ones (CIFAR).

Deep learning for natural language processing R. Socher (Stanford, 2016)

The current trend in natural language processing is to somehow replace words and phrases with vectors, and apply deep learning tools.

For instance, one could take the co-occurrence matrix (from Wikipedia or CommonCrawl), look at the previous and next k words ($k = 5$ to 10 – you can also give more weight to closer words), and reduce the dimension to 25 to 1000, e.g., with SVD. Polysemy, non-symmetry, common words and new words pose problems.

Instead, **Word2vec** tries to predict the previous and next k words

$$P(\text{output} \mid \text{input}) \propto \exp(u'_{\text{output}} v_{\text{input}}).$$

Each word is represented by two vectors, u_w and v_w (concatenate or average them if you want a single one), maximizing the likelihood of the observed data. One can add *negative sampling*: maximize

$$\sigma(u'v) - \sum_k \sigma(-u'_k v)$$

where the u_k are random words, sampled with probability $P(w) \propto U(w)^{3/4}$, where U is the unigram distribution.

The *continuous bag of word* (CBOW) model predicts the center word from the surrounding words.

GloVe uses the (sparse) co-occurrence matrix p and minimizes

$$\sum f(p_{ij})(u'_i v_j - \log p_{ij})^2,$$

where $f(x) = \text{Min}\{x, 100\}$ (to avoid giving too much weight to very common words).

One can predict the sentiment of a word using its context (the vector embedding of $w_{t-2}, w_{t-1}, w_t, w_{t+1}, w_{t+2}$), with a max-margin (hinge) loss, **bidirectional recurrent neural network** (RNN) with LSTM units, clipped gradients,

Use F_1 , the harmonic mean of precision and recall, if the classes are unbalanced.

Recursive neural networks provide embeddings for phrases: their input is the vector embedding of two words (or nodes) and the output is the vector embedding (same dimension) and the score if we merge the two nodes; it can be as simple as one tanh layer, or a different network for each tree depth or phrase type (NP, VP, etc.); it can parse sentences greedily. A PCFG can help reduce the number of candidate trees to examine.

ConvNets are similar to recursive neural nets, but merge all possible trees. Instead of using several layers, use only one, for 2-grams (and 3-gram, 4-grams 5-grams), and add a max-pooling layer, *i.e.*, take the maximum of those n -gram vectors.

DiceKriging, DiceOptim: two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization O. Roustant et al. (JSS, 2012)

Kriging is another name for Gaussian process modeling (the differences lie in the way the trend is modeled: constant, linear, more complex, known, to be estimated, to be integrated out) but it is often limited to 2 or 3 dimensions and Gaussian kernels, which is unsuitable to Bayesian optimization (“computer experiment” is just another name for blackbox (expensive) function).

DiceKriging::km fits a GP model, **DiceOptim::max_EI** returns the next candidate according to the expected improvement (EI) criterion. These functions can be used with **lhs::optimumLHS** for Latin square designs and **sensitivity::fast99** for **sensitivity analysis**.

Given random variables X_1, \dots, X_n , any (L^2) function $f: \mathbf{R}^n \rightarrow \mathbf{R}$ can be decomposed as

$$\begin{aligned} Y &= f(X_1, \dots, X_n) \\ &= \mu_0 + \sum_i \mu_i(X_i) + \sum_{i < j} \mu_{ij}(X_i, X_j) + \dots \\ &= \sum_{I \subset \llbracket 1, n \rrbracket} \mu_I(X_I) \end{aligned}$$

with $E[\mu_I(X_I) | X_J] = 0$ for $J \subsetneq I$, which gives a variance decomposition $\text{Var } Y = \sum_I \text{Var } \mu_I(X_I)$. The *Sobol indices* are $S_I = \text{Var } \mu_I(X_I) / \text{Var } Y$; S_i and $\sum_{I \ni i} S_I$ are of special interest.

Also check: **BACCO**, **mlegp**, **tgp**.

**Optimal combination forecasts
for hierarchical time series**
R.J. Hyndman et al. (2010)

For an additive hierarchy of time series

$$y_t = \sum_i y_{it}, \quad y_{it} = \sum_j y_{ijt}, \quad y_{ijt} = \sum_k y_{ijk t},$$

independent time series forecasts need not be consistent. They are traditionally reconciled, either in a top-down or a bottom-up fashion. Let Y_t be the vector of all the forecasts and Y_{tK} be that of the low-level ones: $Y_t = SY_{tK}$ (where S is a binary matrix). Existing reconciliation methods are of the form $\hat{Y} = SP\hat{Y}$. The minimum variance reconciled estimator solves the optimization problem

$$\begin{array}{ll} \text{Find} & P \\ \text{To minimize} & SP(\text{Var } \hat{Y})P'S' \\ \text{Such that} & SPS = S. \end{array}$$

Equivalently, one can consider the regression $\hat{Y} = S\beta + \varepsilon$, where β are the (unknown, consistent) low-level forecasts.

**The great time series bake off: an experimental
evaluation of recently proposed algorithms**
A. Bagnall et al.

Comparison of time series classification algorithms:

- 1-NN (nearest neighbour) DTW (dynamic time warp) and weighted variants of the DTW distance, e.g., replacing $(x_i - x_j)^2$ with $w(|i - j|)(x_i - x_j)^2$;
- 1-NN for distances using both the series and their first differences;
- Shapelets, *i.e.*, localized patterns (subsequences) used in a decision tree;
- Random-forest based on simple statistics (mean, standard deviation) on subintervals;
- Ensemble methods, in the time, power spectrum, autocorrelation, shapelet, etc. space.

Ensemble methods (and NN DTW) work best.

**Understanding predictive information criteria
for Bayesian models**
A. Gelman et al. (2013)

The AIC and its variants (deviance information criterion, Watanabe-Akaike information criterion) estimate the out-of-sample performance

$$\begin{aligned} \text{AIC} &= -2 \log P(\text{data} | \hat{\theta}_{\text{MLE}}) + 2k \\ \text{DIC} &= -2 \log P(\text{data} | \hat{\theta}_{\text{Bayes}}) + 2p_{\text{Bayes}} \\ \text{WAIC} &= 2 \log P_{\text{post}}(\text{data}) + 2p_{\text{WAIC}} \end{aligned}$$

where

$$\begin{aligned} \hat{\theta}_{\text{Bayes}} &= E_{\text{post}}[\theta] \\ p_{\text{Bayes}} &= 2(\log P(\text{data} | \hat{\theta}_{\text{Bayes}}) - E_{\text{post}}[P(\text{data})]) \\ \text{or } p_{\text{Bayes}} &= 2 \text{Var}_{\text{post}}[\log P(\text{data} | \theta)] \\ \log P_{\text{post}}(\text{data}) &= \sum_i E_{\text{post}}[p(y_i)] \\ p_{\text{WAIC}} &= 2 \sum_i \log E_{\text{post}} p(y_i) - E_{\text{post}} \log p(y_i) \\ \text{or } p_{\text{WAIC}} &= \sum_i \text{Var}_{\text{post}}[\log p(y_i)] \end{aligned}$$

and $E_{\text{post}}[\cdot] = E_{\text{post}}[\cdot | \theta]$, Var_{post} are computed via simulations. (The BIC, $-2 \log P(\text{data} | \hat{\theta}_{\text{MLE}}) + k \log n$, a more conservative measure, was designed to assess model quality, not forecast quality.)

The effective number of parameters in the DIC and the WAIC recognize that hierarchical models have fewer effective than actual parameters.

**Why ℓ^1 is a good approximation to ℓ^0 :
a geometric explanation**
C. Ramirez et al. (2012)

The ℓ^1 norm is the “best” convex relaxation of the ℓ^0 pseudo-norm (or, rather, ℓ^ε , with $\varepsilon > 0$ small): look at natural, exchangeable, archimedean partial orders on \mathbf{R}^n ; they are determined by $B_{\preceq} = \{x : x \preceq (1, 0, \dots, 0)\}$; notice that the ℓ^1 ball $\{\|x\|_1 \leq 1\}$ is the convex hull of the ℓ^0 ball $\{\|x\|_0 \leq 1\}$.

Local ordinal embedding
Y. Terada and U. von Luxburg (2014)

Local ordinal embedding (LOE) recovers the coordinates of a cloud of points from a set of constraints of the form $d_{ij} \leq d_{k\ell}$, $(i, j, k, \ell) \in \mathcal{A}$, by minimizing (by majorization)

$$\sum_{\mathcal{A}} (\delta + d_{ij} - d_{k\ell})_+^2.$$

Local information is sufficient to recover the data. Contrary to other embeddings (t-SNE, etc.), LOE also recovers the density of the data.

**Towards making high-dimensional distance
metric learning practical**
Q. Qian et al.

Distance metric learning (DML) looks for a metric for which points in the same class are close and points in different classes are far away. For high-dimensional problems, one often uses a low-rank parametrization $M = LL'$ or dimension reduction (PCA, random projection) preliminary step. One could look for a positive semi-definite matrix M satisfying (most of the) triplet constraints: for $x \sim y$ and $x \not\sim z$,

$$(x - y)'M(x - y) + 1 \leq (x - z)'M(x - z)$$

i.e., $d(x, y) + 1 \leq d(x, z)$. To find $M \in S_d$ that minimizes

$$\lambda \|M\|_F^2 + \frac{1}{N} \sum_{\substack{x, y, z \\ x \sim y \\ x \not\sim z}} \text{loss}(M, (x-z)(x-z)' - (x-y)(x-y)'),$$

solve the dual; to reduce the dimension, use a random projection (on the dual); to ensure positive semi-definiteness, project onto the positive semidefinite code only once, at the end.

**FaceNet: a unified embedding
for face recognition and clustering**
F. Schroff et al.

To recognize faces regardless of orientation and lighting, train your neural net to learn an embedding where $\|x_1 - x_2\|^2 + \alpha \leq \|x_1 - x_3\|^2$ for all triplets $x_1 \simeq x_2$, $x_1 \not\simeq x_3$; progressively increase the difficulty of the triplets by selecting those with the largest loss in the current minibatch [not unlike boosting].

**A critical review of recurrent neural networks
for sequence learning**
Z.C. Lipton (2015)

Current neural networks can be trained with:

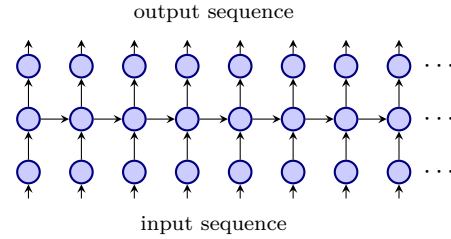
- Newton: $h = -f''(x)^{-1} f'(x)$;
- Approximate Newton: approximate the Hessian, e.g., by assuming it is diagonal, or by updating it every k steps;
- Truncated Newton (Hessian-free): approximate the inverse $f''(x)^{-1}$, e.g. by running conjugate gradient a (very) small number of steps (this does not require the Hessian, just the map $h \mapsto f''(x) \cdot h$);
- Saddle-free: do not rescale the gradient by $1/\lambda_i$ but $1/|\lambda_i|$

and variants of stochastic gradient descent:

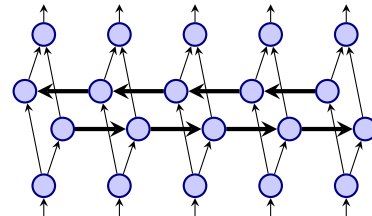
- Momentum: $\text{step}_t \propto \text{gradient} + \alpha \cdot \text{step}_{t-1}$
- Nesterov momentum: idem, but the gradient is taken “half a step” later, i.e., after the momentum step;
- Adagrad: $\text{step}_t \propto \frac{1}{\sqrt{\sum_{k \geq 0} \left| \frac{\partial \text{loss}_{t-k}}{\partial w} \right|^2}}$
- RMSProp: $\text{step}_t \propto \frac{1}{\sqrt{\sum \alpha^k \left| \frac{\partial \text{loss}_{t-k}}{\partial w} \right|^2}}$
- AdaDelta: $\text{step}_t \propto \frac{\sqrt{\sum \alpha^k \|\text{step}_{t-k}\|^2}}{\sqrt{\sum \alpha^k \left| \frac{\partial \text{loss}_{t-k}}{\partial w} \right|^2}}$
- Adam: RMSprop + momentum.

ReLU (restricted linear units) enforce sparsity in the hidden layers.

Recurrent structures are usually of the form:



Bidirectional RNN (BRNN) are of the form:



LSTM (long short term memory) units introduce two changes in the hidden layer, to allow for long memory and avoid vanishing or exploding gradients:

- Self-connected (recurrent) cell with weight 1 (“constant error carousel”);
- Multiplicative nodes, to set or delete the “memory” cell.

**Draw: a recurrent neural network
for image generation**
K. Gregor et al.

Recurrent autoencoders can progressively draw images.

Learning to transduce with unbounded memory
E. Grefenstette et al.

One can devise differentiable analogues of computer elements (e.g., memory cells: LSTM) and data structures (stack, deque, etc.). For instance, a *differentiable stack* is a stack of weighted elements, with weights in $(0, 1)$; the weights decrease when we pop an “element” (but nothing is actually removed); reading the “element on top” returns a weighted sum of the stack elements, with the weights summing to 1, and the higher elements hiding the lower ones as much as possible.

Highway networks
R.K. Srivastava et al.

Use LSTM-like units (gates) in non-recurrent networks to make them deeper but still trainable by SGD: replace $x_{i+1} = H(x_i, w_i)$ with $\lambda = T(x_i, w_i)$, $x_{n+1} = (1 - \lambda)x_i + \lambda H(x_i, w_i)$.

Extremely randomized trees
P. Geurts et al. (2006)

The splits in the trees in a random forest usually come from the data – they can actually be random. [The result can be seen as a smooth and robustified k -NN classifier.]

Deep neural decision forests
P. Kotschieder et al.

Decision trees (and therefore random forests) are amenable to backpropagation: replace the deterministic decision at each node with a stochastic one (a Bernoulli random variable) and define the output of the tree as the expected leaf value.

***Inverse graphics
with probabilistic CAD models***
T.D. Kulkarni et al.

Computer vision is the inverse of computer graphics. It can be approached as a Bayesian inference problem with, as prior, a (random) 3D scene, with lathed objects (using Gaussian processes for their shapes) and human shapes (deformed along their armature), affinely transformed.

Deep convolutional inverse graphics network
T.D. Kulkarni et al.

Computer vision can be seen as the inverse of computer graphics and tackled with an auto-encoder, de-rendering an image into a scene description, and then re-rendering it. The middle layer can be made interpretable by using mini-batches identical except for one aspect (lighting, pose, shape, etc.) and only updating the nodes we want to reflect that aspect.

***Path-SGD: path-normalized optimization
in deep neural networks***
B. Neyshabur et al.

Networks with ReLU units are scale-invariant: multiplying the weights of one layer by λ and dividing those of the next layer by λ leaves the output unchanged. Use scale-invariant optimization algorithms.

Weight uncertainty in neural networks
C. Blundell et al. (2015)

Bayesian neural networks (neural nets whose weights are not numbers but probability distributions) can be seen as a regularization scheme (like early stopping or dropout); in reinforcement learning, they also allow Thompson sampling. A diagonal Gaussian Bayesian neural net (this only doubles the number of parameters) can be trained by back-propagation.

***Evolving neural networks
through augmenting topologies***
K.O. Stanley and R. Miikkulainen (2002)

Genetic algorithms to choose the structure of a neural net.

***MADE:
masked autoencoder for distribution estimation***
M. germain et al. (2015)

An autoencoder can be made autoregressive, *i.e.*, one can ensure that there is no path between input i and output j if $i \geq j$, by using binary mask:

- Connect each hidden node to the first k input nodes, where k is sampled randomly;
- Connect the ℓ th output node to a hidden node if $\ell > k$.

This can be generalized to deeper networks.

***Topological methods
for the analysis of high dimensional data sets
and 3D object recognition***
G. Singh et al. (2007)

Start with a covering $X = \bigcup_{\alpha \in A} U_{\alpha}$; cluster the points in each component, $U_{\alpha} = \bigsqcup_{\beta \in B_{\alpha}} V_{\alpha\beta}$; consider the nerve of the resulting covering $\bar{X} = \bigcup V_{\alpha\beta}$; let the initial covering vary (e.g., balls of radius ε) to have a multiresolution approximation of the dataset.

***Megaman:
manifold learning with millions of points***
J. McQueen et al. (2016)

megaman is an alternative to Scikit-learn's **manifold** submodule, focusing on performance and scalability (sparse matrices, **pyamg**, FLANN, **arpack**, **LOBPCG**, etc.):

- Spectral embedding (Laplacian eigenmaps – fastest);
- Local tangent space adjustment (LTSA)
- Local linear embedding (LLE)
- Isomap
- Diffusion maps

The output also includes an estimate of the distortion.

All those algorithms approximate the manifold with a graph (k -nearest neighbours or, better, ε -radius graphs) and compute its adjacency and/or Laplacian matrix.

***Riemannian preconditioning
for tensor completion***
H. Kasai and B. Mishra

A Tucker decomposition of an order 3 tensor $X \in \mathbf{R}^{n_1 \times n_2 \times n_3}$ is $X = G \times_1 U_1 \times_2 U_2 \times_3 U_3$, where $G \in \mathbf{R}^{r_1 \times r_2 \times r_3}$ and $U_i \in \mathbf{R}^{n_i \times r_i}$ has orthogonal columns. The multilinear rank (r_1, r_2, r_3) of X is the rank of its unfolding matrices (consider X as a cube, cut it into 2-dimensional slices, stack them, and take the rank). Tucker decompositions form a (Stiefel) manifold \mathcal{M} , but they are usually only considered up to rotations: the *Tucker manifold* is $\mathcal{M}/O(r_1) \times O(r_2) \times O(r_3)$.

It can be endowed with a Riemannian metric induced by the Hessian of the loss function of the optimization problem considered, e.g., reconstruction from a small number of entries, under a rank constraint, minimizing the Frobenius norm of the entries available. This is computationally expensive: one can make do with an approximation of the Hessian, assuming that all the entries are available, and only keeping its diagonal blocks.

The usual Riemannian optimization framework then applies.

Fast and guaranteed tensor decomposition via sketching

Y. Wang et al. (2015)

Tensor operations can be sped up by replacing the tensor with a random projection.

Ensemble of exemplar-SVMs for object detection and beyond

T. Malisiewicz

Transfer metadata from the closest exemplar to the observation being classified.

Visualizing large scale and high-dimensional data

J. Tang et al. (2016)

LargeVis is inspired by t-SNE but scales better:

- Compute an approximate k -nearest neighbourhood graph (t-SNE uses an exact one) with a couple of *random projection trees* (pick two points, split the space along their perpendicular bisector hyperplane, iterate) and neighbour exploring (neighbours of neighbours are neighbour candidates) – k -D trees do not work well in high dimensions;
- Use the same weights $p_{j|i} \propto \exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)$ as t-SNE and symmetrize,
- For up to 10^6 points, one can use graph layout algorithms: ForceAtlas2 and Openord are $O(n \log n)$;
- Consider a probabilistic model relating the distance between the points in the low-dimensional projection d_{ij} and the graph weights,

$$\log \text{lik} = \sum_{i-j \text{ edge}} w_{ij} \log f(d_{ij}) - \gamma \sum_{i-j \text{ not edge}} \log(1 - f(d_{ij}))$$

$$f(x) = \frac{1}{1+x^2} \text{ or } \frac{1}{1+e^x}$$

- Replace the $O(n^2)$ negative edges with a smaller number of random edges;
- Optimize with asynchronous stochastic gradient descent.

Unconstrained optimization of real functions in complex variables

L. Sorber et al. (2012)

Traditional optimization algorithms are not directly applicable to real-valued complex functions $\mathbf{C}^n \rightarrow \mathbf{R}$ because they are not (complex) differentiable. Instead of separating the real and complex parts, which would hide some of the structure of the problem, one can use *Wirtinger calculus*, i.e., the complex differentials

$$\frac{\partial}{\partial z} = \frac{\partial}{\partial x} - i \frac{\partial}{\partial y}, \quad \frac{\partial}{\partial \bar{z}} = \frac{\partial}{\partial x} + i \frac{\partial}{\partial y}.$$

Gradient- or Hessian-based algorithms (L-BFGS, CG, Gauss-Newton, Levenberg-Marquardt) then apply.

Recursive decomposition for nonconvex optimization

A.L. Friesen and P. Domingos

Fully decomposable functions, $f(x) = \sum g_i(x_i)$, are easy to minimize, but rare. A function is *globally decomposable* if it can be written $f(x) = g_1(x_0, x_1) + g_2(x_0, x_2)$, where $x = (x_0, x_1, x_2)$ is a partition of the coordinates. A function f is *locally decomposable* if that partition depends on x_0 . It is *approximately locally decomposable* if

$$\|x'_0 - x_0\| \leq \delta \Rightarrow \|f(x') - g_1(x'_0, x'_1) - g_2(x'_0, x'_2)\| \leq \varepsilon.$$

Examples include protein folding (pairwise interactions are negligible if the distance is sufficiently high) or structure from motion (inferring a 3D structure from a set of 2D images). Those problems can be solved by recursive partitioning.

Parallel algebraic modeling for stochastic optimization

J. Huchette et al.

The **StochJuMP** Julia package solves *2-stage stochastic optimization problems with recourse*; contrary to previous implementations, building the model is significantly faster than solving the problem.

Quick introduction to SAT/SMT solvers and symbolic execution

D. Yurichev (2016)

SAT solvers tackle (boolean) satisfiability problems expressed in conjunctive normal form, with ORs ($\bigwedge_i \bigvee_j x_{ij}$, minisat) or XORs ($\bigwedge_i \bigoplus_j x_{ij}$, cryptominisat – XORs are more common than ORs in cryptography).

SMT solvers tackle more general (constraint) satisfiability problems, involving finite sets, integers, reals, arrays, quantifiers. Examples include Z3 (Microsoft, non-free, usable from Python or via a standardized lisp-like language, SMT-LIB), STP, CVC4 (check the SMT competition for a more up-to-date list).

Klee is an LLVM-based symbolic virtual machine, designed to reason about actual programs, generate unit tests, prove code equivalence (e.g., coreutils vs busybox) or correctness, run code backwards – it can also be coaxed into solving more general satisfiability problems.

Klee: unassisted and automatic generation of high-coverage tests for complex systems

D. Dunbar et al.

Symbolic (LLVM) virtual machine, to automatically generate tests or identify bugs.

Faster cover trees

M. Izbicki and C.R. Shelton (2015)

A simplified cover tree (for nearest neighbour queries) is a tree, with one data point in each node, satisfying

$$\begin{aligned}\text{level}(\text{child}) &= \text{level}(\text{parent}) - 1 \\ d(\text{parent}, \text{child}) &\leq 2^{\text{level}(\text{parent})} \\ d(\text{child}, \text{child}_2) &> 2^{\text{level}(\text{child})}\end{aligned}$$

Bloofi: multidimensional Bloom filters

A. Crainiceanu and D. Lemire (2015)

The bitwise OR between Bloom filters (same length, same hash function) is still a Bloom filter. By arranging Bloom filters in a B-tree, one can efficiently find in which Bloom filters an element is.

ZeroDB white paper

M. Egorov and M. Wilkison (2016)

Encrypted databases expose their B-trees to their clients.

A review of homomorphic encryption and software tools for encrypted statistical machine learning **L.J.M. Aslett et al.**

Homomorphic encryption schemes allow some arithmetic operations on the ciphertext, usually additions and multiplications:

$$\text{dec}(\text{enc}(x) \boxplus \text{enc}(y)) = x + y.$$

They currently have the following limitations.

- They typically add noise to a deterministic function, but not too much, to allow for non-ambiguous decryption – but operations on the ciphertext increase that noise. It is possible to reset the level of noise during the computations, but this is complex and slow.
- They are not field morphisms $\mathbf{R} \rightarrow k$; in particular, they do not provide a division: if needed, numerator and denominator can be sent back to the client. Some operations become very expensive: for instance, matrix inversion can be done with cofactors.
- There are no equality or inequality tests: no conditional code flow is possible.
- The ciphertext can be large, – 10,000 times larger than the plaintext. It is possible to use a traditional encryption scheme for the data and homomorphically encrypt the key – but homomorphic AES decryption is very slow.
- Large integers or real numbers have to be encoded, e.g., with the Chinese remainder theorem, or as fractions.

In R, check the `HomomorphicEncryption` package; in C++, check the `HeLib` library.

Encrypted statistical machine learning: new privacy-preserving methods **L.J.M. Aslett (2015)**

The limitations of homomorphic encryption, in particular, the absence of division, exponential and comparisons, requires a redesign of most statistical procedures; the `EncryptedStats` package provides Naives Bayes and extreme random forests. Limited forms of comparison and division are actually available:

- By quantizing the values and replacing them with indicator variables, comparisons such as $x = y$ can be expressed as a scalar product $\sum x_k y_k = 1$; no branching is possible, but $\mathbf{t} ? \mathbf{a} : \mathbf{b}$ can be written $t\mathbf{a} + (1 - t)\mathbf{b}$;
- Approximate division can be performed using (encrypted) random number generation, relying on $X \sim \text{Geom}(p) \implies E[X] = 1/p$.

Smoothing spline ANOVA models: R package gss **C. Gu (JSS, 2014)**

Classical Anova $\mu_{ij} = \mu + \alpha_i + \alpha_j + \varepsilon_{ij}$ can be generalized to functions

$$\begin{aligned}f(x_1, x_2) &= (1 - A_1 + A_1)(1 - A_2 + A_2)f \\ &= A_1 A_2 f + (1 - A_1) A_2 f + (1 - A_2) A_1 f + \\ &\quad (1 - A_1)(1 - A_2)f \\ &= \phi_0 + \phi_1(x_1) + \phi_2(x_2) + \phi_{12}(x_1, x_2)\end{aligned}$$

where A_1 and A_2 are averaging operators, e.g., $(A_1 f)(x_2) = \int_a^b f(x_1, x_2) dx_1 / (b - a)$ and the problem is usually penalized, e.g.,

$$\begin{aligned}\text{Find } & \phi_0, \phi_1, \phi_2, \phi_{12} \\ \text{To minimize } & (\phi_0 - A_1 A_2 f)^2 + \\ & \int (\phi_1 - (1 - A_1) A_2 f)^2 + \\ & \int (\phi_2 - (1 - A_2) A_1 f)^2 + \\ & \iint (\phi_{12} - (1 - A_1)(1 - A_2)f)^2 + \\ & \lambda_1 \int |\phi_1''|^2 + \lambda_2 \int |\phi_2''|^2 + \lambda_{12} \iint \|\phi_{12}''\|^2.\end{aligned}$$

If there are no interactions, this is just a generalized additive model (GAM). The same framework also applies to log-density estimation (with less overfitting than density). In particular, check the `ssanova` and `ssden` functions.

<code>ssanova(y~x)</code>	# Smoothing
<code>ssanova(y~x1+x2)</code>	# GAM
<code>ssanova(y~x1*x2)</code>	# GAM with interactions
<code>ssden(~s)</code>	# Plot with <code>dssden()</code>

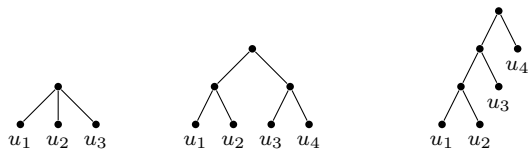
Hierarchical archimedean copulae: the HAC package

O. Okhrin and A. Ristig (JSS, 2014)

Archimedean copulas,

$$C_\phi(u_1, \dots, u_d) = \phi(\phi^{-1}(u_1) + \dots + \phi^{-1}(u_d))$$

are exchangeable. By introducing pseudo-variables $C(u_i, u_j)$, one can consider hierarchical archimedean copulas, e.g.,



$$C(u_1, u_2, u_3) \quad C(C(u_1, u_2), C(u_3, u_4)) \quad C(C(C(u_1, u_2), u_3), u_4)$$

The hierarchical structure can be inferred by finding the two closest variables, replacing them with $C(u_i, u_j)$, and iterating; consecutive nodes with a very similar generator ϕ can be collapsed. Check the `estimate.copula` and `[rpd]HAC` functions.

copulaedas: an R package for estimation of distribution algorithms based on copulas
Y. Gonzalez-Fernandez and M. Soto (2014)

EDA (estimation of distribution algorithms, CMA-ES) optimizes a black-box function as follows:

- Start with a random set of candidates
- Improve them by local search;
- Keep the best ones;
- Estimate their distribution;
- Sample more candidates from this distribution
- Iterate.

The distribution is often estimated as a Gaussian, but one can also use copula-based models: independence copula, Gaussian copula, hierarchical archimedean copula, regular vine, etc.

General purpose convolution algorithm in S4 classes by means of FFT
P. Ruckdeschel and M. Kohl (JSS, 2014)

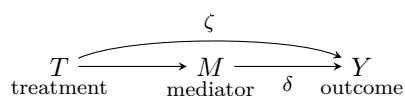
The `distr` package provides operations (+, ×, etc.) on independent random variables, e.g.,

$$X \sim N(0, 3) \text{ and } Y \sim N(0, 4) \implies X + Y \sim N(0, 5)$$

by discretizing and convolving the distributions when no closed form is available.

mediation: R package for causal mediation analysis
D. Tingley et al. (JSS, 2014)

Given random variables T (treatment, binary), M (suspected mediator) and Y (outcome), one can measure how much of a role M plays in the causal process from T to Y



by modeling $M \sim T$ and $Y \sim T + M$ and considering

$$\begin{aligned} \text{Total treatment} \quad & \tau = Y(1, M(1)) + Y(0, M(0)) \\ \text{Causal mediation} \quad & \delta(t) = Y(t, M(1)) + Y(t, M(0)) \\ \text{Direct effect} \quad & \zeta(t) = Y(1, M(t)) + Y(0, M(t)), \end{aligned}$$

which gives decompositions

$$\tau = \delta(t) + \zeta(1 - t), \quad t \in \{0, 1\}.$$

hmm: an R package for hierarchical multinomial marginal models
R. Colombi et al. (JSS, 2014)

Marginal models model the joint distribution of qualitative variables as

$$P(X_1, \dots, X_n) = P(X_1)P(X_2|X_1)P(X_3|X_1, X_2) \cdots P(X_n|X_1, \dots, X_{n-1}),$$

with constraints on some of those factors to enforce conditional independencies (equalities) or stochastic dominance (inequalities) – not unlike graph-less graphical models.

structSSI: simultaneous and selective inference for grouped or hierarchically structured data
K. Sankaran and S. Holmes (JSS, 2014)

Given m p -values $p_1 \leq \dots \leq p_m$, multiple testing procedures usually look at the *family-wise error rate* (FWER), *i.e.*, try to ensure

$$P[\text{at least one false positive}] \leq \alpha$$

- Bonferroni: reject $H_0(k)$ if $p_k \leq \alpha/m$;
- Šidák: reject $H_0(k)$ if $p_k \leq 1 - (1 - \alpha)^{1/m}$;
- Holm–Bonferroni: reject $H_0(k)$ as long as $p_k \leq 1/(m + 1 - k)$;
- Holberg: reject $H_0(k)$ as long as there exists $\ell \geq k$ such that $p_\ell \leq 1/(m + 1 - \ell)$

or the *false discovery rate* (FDR), *i.e.*, try to ensure

$$E \left[\frac{\text{false positives}}{\text{positives}} \right] \leq \alpha$$

- Benjamin–Hochberg (BH): reject $H_0(k)$ as long as $p_k \leq k\alpha/m$;
- BHY: reject $H_0(k)$ as long as $p_k \leq k\alpha / \sum_{i=1}^m (1/i)$.

But they tend to be optimal for independent tests, overly conservative for positively dependent tests, and incorrect for negatively dependent tests – those that remain valid account for an arbitrary dependence structure, but that is too general an assumption.

Some extensions of those tests account for a hierarchical structure:

- Group BH (GBH) estimates the proportion of H_1 's in each group and reweighs the p -values accordingly;
- Hierarchical FDR (HFDR) accepts or rejects the null hypothesis at the group level, and only looks at individual tests in groups whose null hypothesis was rejected.

***LDavis: a method for visualizing
and interpreting topics***
C. Sievert and K.E. Shirley

To find the words most relevant to a given topic, one can rank them according to $P(\text{term}|\text{topic})$ (but this gives too much weight to common, non-discriminative words), or $P(\text{term}|\text{topic})/P(\text{term})$ (lift – but this gives too much weight to rare, non-discriminative words), or, better, a weighted average of their logarithms (“relevance”).

***fitdistrplus:
an R package for fitting distributions***
M.L. Delignette-Muller and C. Dutang (2015)

One can fit a distribution to data using: maximum likelihood (MLE), moment matching, quantile matching, maximum goodness of fit (gof); the gof can be assessed with the Kolmogorov-Smirnov, Cramer-von Mises or Anderson-Darling distance (AD is a variant of CvM with more weight on the tail – there are 1-sided variants, and variants with more weight).

***SDD:
an R package for serial dependence diagrams***
L. Bagnato et al. (JSS, 2015)

The autocorrelation function (ACF) relies on linear correlation. Instead, the *dependogram* uses the independence χ^2 (or its normalization, Cramer’s V) between X and X_{-k} , discretized into n quantile bins, or the Kullback-Leibler divergence between (kernel) density estimators of X , X_{-k} and (X, X_{-k}) .

***Fitting heavy tailed distributions:
the powerLaw package***
C.S. Gillespie (JSS, 2015)

To fit a power law distribution $p(x) \propto x^{-\alpha} \mathbf{1}_{x \geq x_{\min}}$ on the tail of the data, use the maximum likelihood estimator

$$\hat{\alpha} = 1 + n \left/ \sum_{x \geq x_{\min}} \log \frac{x}{x_{\min}} \right/;$$

choose the x_{\min} minimizing the Kolmogorov-Smirnov (KS) distance; bootstrap to estimate the uncertainty on x_{\min} ; to estimate the goodness of fit, look at the distribution of the KS statistic on data sampled from \hat{p} .

***abctools: an R package for tuning
approximate Bayesian computation analyses***
M.A. Nunes and D. Prangle

Approximate Bayesian computation (ABC) is a Bayesian method that uses simulations instead of likelihoods. Rejection ABC proceeds as follows:

- Draw a parameter θ from the prior;
- Simulate data $x \sim p(\cdot|\theta)$; compute statistics $s(x)$;
- If $d(s^{\text{obs}}, s) \leq \varepsilon$, accept θ ;
- Iterate until you have enough samples.

The *abctools* package complements the *abc* package and can help choose the statistics from a set of candidates, e.g., by approximate sufficiency (greedily add the statistics that produce the largest change in the posterior distribution; stop when it becomes insignificant) or by finding the subset minimizing some information criterion (there are also projection methods) and help choose ε .

***ngspatial: a package for fitting the centered
autologistic and sparse spatial generalized
linear mixed models for areal data***
J. Hughes

The autologistic model

$$\log \frac{P[z_i = 1]}{P[z_i = 0]} = x'_i \beta + \eta \sum_{j \in \text{Neigh}(i)} Z_j$$

is confounded; one can center the autocovariate with

$$\log \frac{P[z_i = 1]}{P[z_i = 0]} = x'_i \beta + \eta \sum_{j \in \text{Neigh}(i)} (Z_j - \mu_j)$$

$$\mu_j = (1 + e^{-x_j \beta})^{-1}.$$

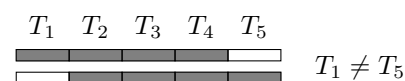
***MVN: an R package
for assessing multivariate normality***
S. Korkmaz et al.

To check multivariate normality:

- Look at the 1-dimensional margins: histograms, qq-plots, normality tests;
- Plot the 2-dimensional densities (*persp*, *contour*);
- Plot the sample Mahalanobis distance versus χ^2 quantiles;
- Try multivariate normality tests: Mardia (based on skewness and kurtosis), Henze-Zirkler (combines $\sum e^{-\alpha D_{ij}}$ and $\sum e^{-\beta D_i}$, where $D_{ij} = d_{\text{Mahalanobis}}(x_i, x_j)$ and $D_i = d_{\text{Mahalanobis}}(x_i, \bar{x})$, Royston (Shapiro-Wilk);
- Remove outliers (robust Mahalanobis distance, from minimum covariance determinant estimators instead of sample covariances).

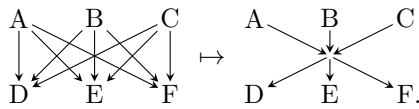
***paircompviz: and R package for visualization
of multiple pairwise comparison test results***
M. Burda

The result of multiple pairwise tests is often represented by a line diagram, or a letter diagram.



Instead, one can consider the relation $T_i \succ T_j$ if i is significantly different from j , hope it is a partial order (this happens surprisingly often) and draw its *Hasse diagram* (i.e., remove the edges that can be inferred

by connectivity) after compressing complete bipartite subgraphs



logcondens: computation related to univariate log-concave density estimation
L. Dümbgen and K. Rufibach

The log-concave density estimator is parameter-free, and log-concavity is a plausible assumption for many datasets: consider using its smoothed version (it is piecewise log-linear and has discontinuities at both ends of the support) instead of a kernel density estimator.

For a multivariate estimator, check LogConcDEAD.

Online graph pruning for pathfinding on grid maps
D. Harabor and A. Grastien

To account for symmetry and avoid examining nodes unnecessarily, path finding algorithms on grids try to recognize rectangular rooms, dead-ends, swamps, neighbours to prune and “jump points”.

Metabolic paths in world economy and crude oil price
F. Picciolo et al. (2015)

Measure paths in the world trade web (WTW) with:

- Trade imbalance $b = \frac{\sum_i \text{Min}(w_{\cdot i}, w_{i \cdot})}{\sum_{ij} w_{ij}}$
- Reciprocity $r = \frac{\sum_{ij} \text{Min}(w_{ij}, w_{ji})}{\sum_{ij} w_{ij}}$
- Cycling index

$$\Gamma^{(s)} = \frac{\sum s_i \Gamma_i^{(s)}}{\sum w_i}$$

$$\Gamma_i^{(s)} = \frac{u_{ii}^{(s)} - 1}{u_{ii}^{(s)}}$$

$$U^s = I + M + M^2 + \dots + M^s$$

$$m_{ij} \propto w_{ij} \text{ (stochastic matrix)}$$

where $\Gamma_i^{(s)}$ is the fraction of trade that goes back to i within s steps.

Look at the corresponding time series, the contribution of each country, the correlation with oil, etc.

Forecasting stock returns during good and bad times
D. Huang et al. (2012)

Log-prices can be modeled as the sum of a random walk (permanent shocks) and an AR(1) process (temporary

shocks)

$$\begin{pmatrix} p_t \\ q_t \\ z_t \end{pmatrix} = \begin{pmatrix} 0 \\ \mu \\ 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & \lambda \end{pmatrix} \begin{pmatrix} p_{t-1} \\ q_{t-1} \\ z_{t-1} \end{pmatrix} + \varepsilon$$

$$\text{Var } \varepsilon = \begin{pmatrix} 0 & 0 & 0 \\ 0 & \sigma_1^2 & \rho \sigma_1 \sigma_2 \\ 0 & \rho \sigma_1 \sigma_2 & \sigma_2^2 \end{pmatrix}$$

where the correlation ρ between the random walk and the AR(1) innovations is a market state indicator: $\rho \geq 0$ corresponds to good times, overconfidence, low risk premium, reversal, $\text{Cov}(r_{t+1}, r_t) < 0$. Empirically, use $\text{MRI} = (12\text{-month return} - \mu)/\sigma$, (where μ and σ are the long-term mean and standard deviation of the 12-month return) as momentum, and $\text{IMA} = \mathbf{1}_{\log\text{-price} > \text{MA}(\log\text{-price}, 200d)}$ or $\text{ISR} = \mathbf{1}_{\text{MA}(\text{Sharpe}, 6m) > 30\% \text{ long-term quantile}}$ as market state indicator.

Risk premia: asymmetric tail risks and excess returns
Y. Lempérière et al.

The *ranked PEL skewness* is the area under the curve of the cumulated standardized values, sorted by increasing absolute values (there are other low-moment estimators of skewness, e.g., the normalized mean-minus-median). The risk premium present in many markets seems to be due to skewness rather than volatility – investors do not fear small Gaussian fluctuations, but rather large tail events.

Forecasting the equity risk premium: the role of technical indicators
C.J. Neely et al. (2010)

To forecast index returns, use both macroeconomic variables (dividend yield, robust excess return volatility, term spread) and technical indicators (MACD, momentum, MACD of the signed volume).

Financial volatility and economic activity
F. Fornari and A. Mele (2010)

Volatility (robust MAD estimator, $\sqrt{\frac{\pi}{2}} \langle |\text{ret}_t| \rangle$) and term spread can predict industrial production growth.

Macroeconomic determinants of stock volatility and volatility premiums
V. Corradi et al. (2012)

Conversely, the business cycle can predict volatility.

Short interest and aggregate stock returns
D.E. Rapach (2014)

Aggregate short interest (as a proportion of shares outstanding, detrended) is informative.

The vector algebra war: a historical perspective J.M. Chappell

Different formulations can be used to compactly write Maxwell's field equations:

- Gibbs vectors, *i.e.*, vectors in \mathbf{R}^3 with dot and cross product $(\mathbf{R}, \cdot, \times)$;
- Quaternions \mathbf{H} , a 4-dimensional algebra with $i^2 = j^2 = k^2 = ijk = -1$;
- Clifford algebra, $\mathcal{Cl}(\mathbf{R}^3)$, an 8-dimensional algebra, with $e_i e_j = -e_j e_i$, $e_1^2 = e_2^2 = e_3^2 = +1$, with the natural embedding $e_i \mapsto e_i$ from \mathbf{R}^3 ; the dot and cross product can be recovered as $vw = v \cdot w + e_1 e_2 e_3 v \times w$.

With the Clifford (or geometric) algebra, Maxwell's equations become $\partial(E + jcB) = \rho/\varepsilon - \mu cJ$. [No mention of differential forms or screw calculus.]

Recent results on pattern maximum likelihood J. Acharya et al. (2009)

Given an iid sequence of symbols, e.g., HHTHTTH, where the list of possible symbols is not known, what can we say about the multiset of symbol probabilities? Standard maximum likelihood (estimate the probabilities of each symbol and forget the order) only works well for large samples. *Pattern maximum likelihood* (PML) replaces the symbols by numbers, starting with the most frequent, and reorders them, e.g., ababca \rightarrow 121231 \rightarrow 111223. A pattern probability p is a non-increasing sequence $p_1 \geq p_2 \geq \dots \geq 0$ with $\sum p_i \leq 1$; the remainder $1 - \sum p_i$ is the continuous part of p . The PML of a pattern ψ is

$$\text{PML}(\psi) = \underset{p}{\text{Argmax}} p(\psi).$$

Algebraic computation of pattern maximum likelihood J. Acharya et al. (2011)

The PML can be computed analytically with Gröbner bases. For instance, $\text{PML}(1112234) = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$.

Analytic combinatorics P. Flajolet and R. Sedgewick (2009)

The traditional approach to combinatorics, to estimate the number a_n of a certain type of objects (e.g., graphs of size n with a certain property) is the following:

- Find a recurrence relation involving a_n (if you can solve it, you are done);
- Consider the generating function $f(x) = \sum a_n x^n$ or (if a grows too fast) $f(x) = \sum a_n x^n / n!$;
- Transform the recurrence relation for a into a functional (or differential) equation for f ;
- Solve the equation or, somehow, use it to infer the asymptotic behaviour of a .

Analytic combinatorics goes directly from the combinatorial description of the problem to the functional equation for the generating function, and uses com-

plex analysis to estimate the asymptotic behaviour of its coefficients.

1. A *combinatorial class* $(\mathcal{A}, |\cdot|)$ is a set \mathcal{A} with a size function $|\cdot| : \mathcal{A} \rightarrow \mathbf{N}$ such that, for all n , $a_n = \#\{x \in \mathcal{A} : |x| = n\}$ is finite. Its *ordinary generating function* is $A(z) = \sum a_n z^n \in \mathbf{R}[[z]]$. The coefficients are denoted $[z^n]A(z) = a_n$.

Simple operations on combinatorial classes translate to their generating functions as follows.

$\mathcal{A} \sqcup \mathcal{B}$	$C(z) = A(z) + B(z)$
$\mathcal{A} \times \mathcal{B}$	$C(z) = A(z)B(z)$
$\text{SEQ } \mathcal{B}$	$C(z) = 1/(1 - B(z))$
$\text{CYC } \mathcal{B}$	$C(z) = \sum_{k \geq 1} \frac{\phi(k)}{k} \log \frac{1}{1 - B(z^k)}$
$\text{MSET } \mathcal{B}$	$C(z) = \prod_{n \geq 1} (1 - z^n)^{-B_n}$
	$= \exp \left(\sum_{k \geq 1} \frac{1}{k} B(z^k) \right)$
$\text{PSET } \mathcal{B}$	$C(z) = \prod_{n \geq 1} (1 + z^n)^{B_n}$
	$= \exp \left(\sum_{k \geq 1} \frac{(-1)^{k-1}}{k} B(z^k) \right)$

The last three formulas are the Pólya logarithm, exponential, and modified exponential. The sequences $\text{SEQ } \mathcal{B} = \{\varepsilon\} \sqcup \mathcal{B} \times \mathcal{B} \times \mathcal{B} \times \dots = \coprod_{n \geq 0} \mathcal{B}^n$ are only defined if $b_0 = 0$; the cycles are $\text{CYC } \mathcal{B} = (\text{SEQ } \mathcal{B} \setminus \{\varepsilon\})/\mathfrak{S}$; the multisets are $\text{MSET } \mathcal{B} = (\text{SEQ } \mathcal{B})/\mathfrak{S}$.

Here are a few examples.

- E : single object of size 0;
- Z : single object of size 1;
- $I = \text{SEQ}_{\geq 1} Z$: non-zero integers;
- $\text{SEQ}(Z \sqcup Z \times Z)$: coverings of $\llbracket 0, n \rrbracket$ by intervals of length 1 or 2 (Fibonacci numbers);
- $T = E \sqcup T \times Z \times T$: triangulations (Catalan numbers);
- $G = Z \times \text{SEQ } G$: rooted plane trees – a tree is a root and a sequence of trees (shifted Catalan numbers shifted);
- $H = Z \times \text{MSET } H$: non-plane (rooted) trees;
- $C = \text{SEQ } I$: compositions (ways of writing an integer as a sum of integers);
- $P = (\text{SEQ } I)/\mathfrak{S} = \text{MSET } I$: partitions;
- $A = mZ$: m -letter alphabet;
- $W = \text{SEQ } A$: words.

The language recognized by a finite automaton has generating function $L(z) = \mathbf{u}'(I - zT)\mathbf{v}$, where T is the incidence matrix (it is not binary because there can be several edges, with different labels, between the same vertices), and \mathbf{u} and \mathbf{v} the indicator vectors of the initial and final states. Rational languages have a rational generating function; context-free languages have an algebraic generating function.

2. A *labelled combinatorial class* is a combinatorial class $(\mathcal{A}, |\cdot|)$ with an action of the symmetric group \mathfrak{S}_n on each $\mathcal{A}_n = \{x \in \mathcal{A} : |x| = n\}$. Informally, think of the elements of \mathcal{A} as graphs (perhaps with some added structure) whose vertex set is $[1, n]$ – the notion of *species of structure* makes this rigorous and unifies the labelled and unlabelled cases. Its *exponential generating function* is $A(z) = \sum a_n x^n / n!$. Operations on labelled combinatorial classes translate to their exponential generating functions.

$$\begin{array}{ll} \mathcal{A} \sqcup \mathcal{B} & C(z) = A(z) + B(z) \\ \mathcal{A} \star \mathcal{B} & C(z) = A(z)B(z) \\ \text{SEQ } \mathcal{B} & C(z) = 1/(1 - B(z)) \\ \text{SET } \mathcal{B} & C(z) = \exp B(z) \\ \text{CYC } \mathcal{B} & C(z) = \log \frac{1}{1 - B(z)} \end{array}$$

Given two labelled objects $\alpha \in \mathcal{A}$, $\beta \in \mathcal{B}$, one can build labelled objects (α', β') by relabelling α and β while keeping their order (there are many such relabelings). The set of all such labelled objects form the *labelled product* $\mathcal{A} \star \mathcal{B}$.

Examples include:

- $R = \text{SEQ SET}_{\geq 1} Z$: surjections;
- $S = \text{SET SEQ}_{\geq 1} Z$: partitions;
- $P = \text{SET CYC} Z$: permutations;
- $I = \text{SET CYC}_{1,2} Z$: permutations;
- $D = \text{SET CYC}_{>1} Z$: derangements;
- $T = Z \star \text{SET } T$, $T(z) = ze^{T(z)}$: a (non-plane) rooted tree is a root and a set of rooted trees;
- $T = U^\bullet$: a rooted tree is a pointed unrooted tree;
- $F = \text{SET CYC } T$, $F(z) = (1 - T(z))^{-1}$: the graph of an endomorphism is a set of cycles, with a tree attached to each element of those cycles.

3. This can be generalized to combinatorial classes with parameters $(\mathcal{A}, |\cdot|, \chi)$, $\chi : \mathcal{A} \rightarrow \mathbf{N}^d$, and multivariate generating functions, $A(z, \mathbf{u}) = \sum a_{n, \mathbf{k}} z^n \mathbf{u}^{\mathbf{k}}$, where $a_{n, \mathbf{k}} = \#\{a \in \mathcal{A} : |a| = n \text{ and } \chi(a) = \mathbf{k}\}$. This leads, for instance, to the expected number of cycles in a permutations of size n , $\mu_n = \sum_{k=1}^n 1/k \sim \log n$.

4. Complex analysis, starting with Cauchy's formula

$$[z^n]f(z) = \frac{1}{2\pi i} \oint_{\Gamma} \frac{f(z)dz}{z^{n+1}},$$

can help study the asymptotic behaviour of those power series.

If f is analytic at 0, with radius of convergence R , then $f_n = [z^n]f(z)$ is of exponential order R^{-n} , $f_n \asymp R^{-n}$, i.e., $\limsup |a_n|^{1/n} = 1/R$.

Saddlepoint bounds are often accurate:

$$[z^n]f(z) \leq \inf_{r \in (0, R)} r^{-n} \sup_{|z|=r} |f(z)|.$$

If f is meromorphic on $[|z| = R]$, then

$$[z^n]f(z) = \sum P_i(n) \alpha_i^{-n} + O(R^{-n}),$$

where α_i are the poles of f on the circle $[|z| = R]$ and P_i are polynomials of degree at most the order of the poles minus 1.

6. If f is meromorphic on $D = [|z| \leq R]$, with radius of convergence R , and dominant singularities (i.e., on the boundary of the convergence disk) ζ_1, \dots, ζ_k , then $g(z) = f(z) - \sum c_i (1 - z/\zeta_i)^{-\alpha_i}$ is holomorphic on a neighbourhood of D , and the asymptotic behaviour of $[z^n]f(z)$ is that of $\sum c_i (1 - z/\zeta_i)^{-\alpha_i}$. For a more precise asymptotic estimate, repeat the process with g .

In particular, if there is only one dominant singularity,

$$f(x) \sim (1 - z/\zeta)^{-\alpha} \implies f_n \sim \frac{1}{\Gamma(\alpha)} \rho^{-n} n^{\alpha-1}.$$

This can be generalized to singularities of the form

$$(1 - z)^{-\alpha} \left(\frac{1}{z} \log \frac{1}{1 - z} \right)^\beta.$$

8. If there are no singularities, or if the closest singularity is essential, the saddle point method can give the asymptotic behaviour of the coefficients.

Given an analytic function f on $\Omega \subset \mathbf{C}$, the surface $z \mapsto |f(z)|$ is not arbitrary: its points are either ordinary ($f \neq 0$, $f' \neq 0$), zeroes ($f = 0$) or saddle points ($f \neq 0$, $f' = 0$) – in particular, the zeroes are the only extrema (this can be used to prove the fundamental theorem of algebra).

A trivial bound for the integral $\int_A^B f(z)dz$ is

$$\left| \int_A^B f(z)dz \right| \leq \text{length}(\Gamma) \times \sup_{\Gamma} |f|.$$

To have a tighter bound, one can choose a path Γ that goes through a saddle point. In particular (if G is analytic at zero, not a polynomial, with nonnegative coefficients and $G(R) = +\infty$),

$$[z^n]G(z) = \frac{1}{2\pi i} \oint \frac{G(z)dz}{z^{n+1}} \leq \frac{G(\zeta)}{\zeta^n}$$

where $\zeta G'(\zeta)/G(\zeta) = n + 1$ (the saddle point, ζ , and the circle of integration depend on n).

The *saddle point method* is a refinement of this bound using the Laplace method: (under a few assumptions)

$$\begin{aligned} [z^n]G(z) &\sim \frac{G(\zeta)}{\zeta^n \sqrt{2\pi b(\zeta)}} \\ b(z) &= z^2 \frac{d^2}{dz^2} \log G(z) + z \frac{d}{dz} \log G(z) \\ \zeta \frac{G'(\zeta)}{G(\zeta)} &= n. \end{aligned}$$

9. The multivariate power series $F(x, u) = \sum f_{n, k} u^k x^n$ can be seen as a deformation of a univariate series $F(x, 1)$. For n fixed, it gives the probability generating function of a discrete distribution

$$f_n^{-1} [x^n] F(x, u) = f_n^{-1} \sum_k f_{n, k} u^k$$

and can help study the limit law, the convergence speed (Berry-Esseen) and the tail probabilities.

[For discrete laws, if $p_n(u) = E[u_n^X]$ pointwise converges to $p(u) = E[u^X]$ for all $u \in \mathbf{C}$, where A has an accumulation point in the open unit disk, then X_n converges to X in distribution. For continuous laws ($\mu_n, \sigma_n \rightarrow \infty$ – rescaling is needed), pointwise convergence of the characteristic function (on \mathbf{R}) or of the Laplace function (on a neighbourhood of 0) implies convergence in distribution.]

Under some conditions (involving the Laplace transform, but reducing to easy-to-check conditions depending on the type of singularity), the distribution is asymptotically Gaussian.

Introduction to the theory of species of structures F. Bergeron et al. (2013)

A *species of structures* is a functor F from the category of finite sets and bijections \mathbf{set} to itself.

To a species F , one can associate formal power series

$$F(x) = \sum_{n \geq 0} \#F[n] \frac{x^n}{n!}$$

$$\tilde{F}(x) = \sum_{n \geq 0} \#(F[n]/\simeq) x^n$$

$$Z_F(x_1, \dots) = \sum_{n \geq 0} \frac{1}{n!} \sum_{\sigma \in \mathfrak{S}_n} \text{fix } F[\sigma] x_1^{\sigma_1} x_2^{\sigma_2} \dots$$

where $[n] = \{1, 2, \dots, n\}$, $\#(F[n]/\simeq) = \#(F[n]/\mathfrak{S}_n)$ is the number of isomorphism classes in $F[n]$, σ_k is the number of cycles of length k in the cycle decomposition of $\sigma \in \mathfrak{S}_n$, $\text{fix } F[\sigma] = \#\{u \in F[n] : \sigma u = u\}$ is the number of points of $F[n]$ fixed by $F[\sigma]$.

The *exponential generating series* $F(x)$ counts (labelled) F -structures; the (isomorphism) *type generating series* $\tilde{F}(x)$ counts isomorphism classes of F -structures, i.e., unlabelled F -structures; the *cycle index series* Z_F contains more information.

Bijections $FU \simeq GU$ for all U (equipotence) do not necessarily define an isomorphism: for instance, the species of linear orders Lin and that of permutations Perm are equipotent, but not isomorphic (the bijections do not form a natural transformation); in particular, $\text{Lin}(x) = \text{Perm}(x)$, but $\widetilde{\text{Lin}}(x) \neq \widetilde{\text{Perm}}(x)$.

The following operations on species correspond to more

or less complex operations on their power series.

Sum	$(F + G)U = FU \sqcup GU$
Product	$(F \cdot G)U = \sum_{U=U_1 \sqcup U_2} FU_1 \times GU_2$
Substitution	$(F \circ G)U = \sum_{\pi \text{ partition of } U} F[\pi] \times \prod_{p \in \pi} G[p]$
Derivative	$F'U = F(U \sqcup \{*\})$
Pointing	$F \bullet U = F[U] \times U$
Superposition	$(F \times G)U = FU \times GU$
Composition	$(F \square G)U = F(GU)$

Here are a few examples.

A permutation can be decomposed into cycles: the cycles of length greater than 1 form a derangement, the cycles of length 1 form a set. This corresponds to a product of species, $\text{Perm} = \text{Der} \cdot U$, where $U : E \mapsto \{*\}$ is the species of sets, and gives $1/(1-x) = \text{Der}(x)e^x$.

$$\text{Perm}(E) = \coprod_{U=E_1 \sqcup E_2} \text{Der}(E_1) \times U(E_2)$$

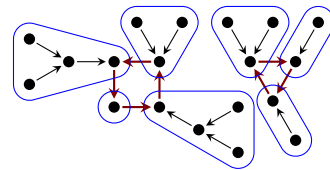
Since a map is a surjection onto its image, we can relate maps and surjections by $\text{Set}(I, \cdot) = \text{Epi}(I, \cdot) \cdot U$:

$$\text{Set}(I, J) = \coprod_{J=J_1 \sqcup J_2} \text{Epi}(I, J_1) \times U(J_2).$$

A map $E \rightarrow E$ is defined by (iterate the map until you get cycles, and look at the basin of attraction of each element in those cycles):

- A partition π of E ;
- A rooted tree on each block of π ;
- A permutation on the blocks of π .

This is a composition of species, $\text{End} = \text{Perm}(A)$.



A partition of E is:

- A partition π of E ;
- A set structure on each block of π ;
- A set structure on π .

This gives $\text{Partitions} = U(U-1)$, and $\text{Partitions}(x) = \exp(e^x - 1)$ (Bell numbers).

A *weighted species* is a functor $\mathbf{set} \rightarrow R\text{-}\mathbf{set}$, where $R = k((t_1, t_2, \dots))$ is a field of formal power series and $R\text{-}\mathbf{set}$ is the category of R -weighted finite sets. An R -weighted finite set is a set A with a map $w : A \rightarrow R$. One associates power series associated to a weighted species as before, by replacing the size $|A|$ with the inventory $|A|_w = \sum_{a \in A} w(a)$. The usual operations $(+, \cdot, \circ, ', \bullet, \times, \square)$ extend to weighted species. Weighted

species can be used to compute, e.g., the number of rooted trees with a given number of nodes and a given number of leaves.

A k -sort species is a functor $\mathbf{mset}(k) \rightarrow \mathbf{set}$, where $\mathbf{mset}(k)$ is the category of multisets with k sorts of elements (colours), *i.e.*, of k -tuples of sets.

Qu'est-ce qu'une espèce de structures?
Génèse et description
F. Bergeron and G. Labelle (2011)

More elementary introduction to the theory of species of structures.

*Statistical learning with sparsity:
the lasso and its generalizations*

T. Hastie, R. Tibshirani, M. Wainwright (2015)

2. The lasso

$$\begin{array}{ll} \text{Find} & \beta \\ \text{To minimize} & \frac{1}{2N} \|y - X\beta\|_2^2 \\ \text{Such that} & \|\beta\|_1 \leq t \end{array}$$

can be put in *Lagrangian* form

$$\text{Argmin}_{\beta} \frac{1}{2N} \|y - X\beta\|_2^2 + \lambda \|\beta\|_1.$$

The $1/2N$ coefficient makes models of different sizes comparable (e.g., for cross-validation).

For a single predictor, the fit can be obtained by *soft-thresholding* the least squares estimate

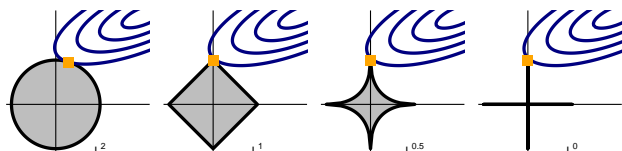
$$S_{\lambda}(\beta) = \text{sign}(\beta)(|\beta| - \lambda)_+ = \text{soft}(\beta, \lambda).$$

Cyclical *coordinate descent* updates the coefficients one at a time, keeping the others fixed, by soft-thresholding.

The *degrees of freedom* of an estimator for an additive error model $y_i = f(x_i) + \varepsilon_i$, f unknown, $\varepsilon_i \sim N(0, \sigma^2)$, is

$$\text{df}(\hat{y}) = \frac{1}{\sigma^2} \sum_i \text{Cov}(\hat{y}_i, y_i).$$

One could also consider ℓ^p penalties or constraints, with $0 \leq p < 1$, but the problem would no longer be convex.



3. Generalized linear models (logistic, survival, Poisson, etc.) are estimated by minimizing the log-likelihood; one can simply add an ℓ^1 penalty to them.

Thanks to the ℓ^1 penalty, overspecified models are not a problem: there is no need to encode factors as “contrasts”.

The ℓ^1 regularized SVM path presents jumps: to avoid them, replace the hinge loss with its square; it becomes very similar to logistic regression.

$$\text{Argmin}_{\beta} \frac{1}{N} \sum (1 - y_i f_i(x_i|\beta))_+^2 + \lambda \|\beta\|_1.$$

4. The elastic net combines ℓ^1 and ℓ^2 penalties. An ℓ^1 penalty alone does not work well when there are identical variables (the solution is not unique).

To have groups of variables enter the model or remain unselected in block, the **group lasso** adds their unsquared ℓ^2 norm as a penalty. Larger groups are more likely to be selected: to avoid that, weigh the penalties with the group size, e.g., \sqrt{p} or (better) $\|Z\|_F$ (Frobenius norm of the design matrix of those factors).

To enforce sparsity inside each group, the *sparse group lasso* adds an ℓ^1 penalty in addition to the unsquared ℓ^2 one.

Sometimes, those groups of variables may overlap (e.g., gene pathways). The *overlap group lasso* can also enforce a hierarchical structure, e.g., to allow interactions only if the main effects are already present.

Generalized additive models (GAM) are usually estimated by backfitting, $f_j \leftarrow \text{Smooth}(y - \sum_{k \neq j} \hat{f}_k)$; they solve the optimization problem

$$(\hat{f}_1, \dots, \hat{f}_J) = \text{Argmin}_{f_1, \dots, f_J} E \left[Y - \sum_j f_j(X_j) \right]^2.$$

Sparse GAM (SPAM) adds an unsquared ℓ^2 penalty and uses sparse backfitting:

$$\text{Argmin}_{f_1, \dots, f_J} E \left[Y - \sum_j f_j(X_j) \right]^2 + \lambda \sum_j \|f_j\|_2$$

$$\tilde{f}_j \leftarrow \text{Smooth} \left(y - \sum_{k \neq j} \hat{f}_k \right)$$

$$\hat{f}_j \leftarrow \left(1 - \frac{\lambda}{\|\tilde{f}_j\|_2} \right)_+ \tilde{f}_j$$

The **fused lasso** approximates a time series with a piecewise constant function

$$\text{Argmin}_{\theta} \frac{1}{2} \sum (y_i - \theta_i)^2 + \lambda \sum |\theta_i - \theta_{i-1}|.$$

It can be generalized to images, non-equispaced observations, regression with ordered predictors ($|\beta_i - \beta_{i-1}|$ – nearby predictors tend to have the same coefficient), piecewise affine or piecewise polynomial functions (*trend filtering*), nearly isotonic regression $\|(\Delta\theta)_-\|_1$.

Coordinate descent does not work for the fused lasso. In the 1-dimensional case, one can start with the unpenalized estimate ($\lambda = 0$) and progressively fuse neighboring intervals. One can also solve the dual (lifted) problem or use dynamic programming.

5. The *nuclear norm* (the sum of the singular values) is a generalization of the ℓ^1 norm: $\|\text{diag } x\|_* = \|x\|_1$. It is also a convex relaxation of the rank.

Descent methods are iterative optimization algorithms: choose a direction (usually not far from the gradient, i.e., $\langle \nabla f, \text{dir} \rangle < 0$); make a small step in this direction; iterate. Common direction choices include

$$\begin{aligned}\text{dir} &= -\nabla f && \text{(gradient descent)} \\ \text{dir} &= -D^{-1}\nabla f && \text{for some diagonal matrix } D \\ \text{dir} &= -(\nabla^2 f)^{-1}\nabla f && \text{(Newton).}\end{aligned}$$

The step size is important: estimate it with a 1-dimensional optimization, or with Armijo's rule.

In the presence of constraints, the *projected gradient* method simply projects the solution back onto the feasible set after each step (gradient descent is similar, with $\beta \in \mathbf{R}^n$ instead of $\beta \in C$):

$$\beta_{n+1} = \underset{\beta \in C}{\text{Argmin}} f(\beta_n) + \langle \nabla f(\beta_n), \beta - \beta_n \rangle + \frac{1}{2s} \|\beta - \beta_n\|_2^2.$$

If the objective is a sum of convex functions $f = g + h$, one differentiable, one not, the *generalized gradient update* linearizes g but leaves h as is.

$$\begin{aligned}\beta_{n+1} &= \underset{\beta}{\text{Argmin}} g(\beta_n) + \langle \nabla g(\beta_n), \beta - \beta_n \rangle + \\ &\quad \frac{1}{2s} \|\beta - \beta_n\|_2^2 + h(\beta) \\ &= \text{prox}_{sh}(\beta_n - s\nabla g\beta_n)\end{aligned}$$

The **proximal** map generalizes the projection:

$$\begin{aligned}\text{prox}_h z &= \underset{\theta}{\text{Argmin}} \frac{1}{2} \|z - \theta\|_2^2 + h(\theta) \\ \text{prox}_{I_C} z &= \text{projection of } z \text{ onto } C \\ \text{prox}_{\lambda\|\cdot\|_1} z &= S_\lambda(z) \text{ elementwise soft-thresholding} \\ \text{prox}_{\|\cdot\|_*} Z &= \text{singular value soft-thresholding.}\end{aligned}$$

Convergence is faster if g is strongly convex.

The accelerated gradient descent, e.g., conjugate gradient, or **Nesterov momentum**

$$\begin{aligned}\beta_{n+1} &= \theta_n - s\nabla f(\theta_n) \\ \theta_{n+1} &= \beta_{n+1} + \frac{n}{n+1}(\beta_{n+1} - \beta_n)\end{aligned}$$

is faster (but non-monotonic).

Coordinate descent (optimizing one coordinate at a time) works if the non-differentiable part of f is separable or, more generally if, when the directional derivatives along the axes are non-negative, then so are all the directional derivatives (for a situation where it does not happen, take f on \mathbf{R}^2 with polygonal level curves: coordinate descent cannot escape (acute) corners that remain inside one of the quadrants).

For linear regression, if the variables are centered and standardised, the lasso and the elastic net estimates

can be computed from the OLS estimates by shrinkage and soft-thresholding.

Coordinate descent is faster than the proximal gradient or Nesterov momentum.

Least angle regression (LAR) adds the variables one at a time but, contrary to stepwise regression, it only increases the coefficients until another variable becomes more promising. If the algorithm also removes a variable from the active set when its coefficient drops to zero, it computes the lasso regularization path.

For the alternating direction method of multipliers (ADMM),

$$\begin{aligned}\text{Find} &\quad \beta, \theta \\ \text{To minimize} &\quad f(\beta) + g(\theta) \\ \text{Such that} &\quad A\beta + B\theta = c\end{aligned}$$

the augmented Lagrangian is (with g small)

$$\begin{aligned}L_\rho(\beta, \theta, \mu) &= f(\beta) + g(\theta) + \langle \mu, A\beta + B\theta - c \rangle + \\ &\quad \frac{\rho}{2} \|A\beta + B\theta - c\|_2^2.\end{aligned}$$

It can be solved by iterating

$$\begin{aligned}\beta_{n+1} &= \underset{\beta}{\text{Argmin}} L_\rho(\beta, \theta_n, \mu_n) \\ \theta_{n+1} &= \underset{\theta}{\text{Argmin}} L_\rho(\beta_{n+1}, \theta, \mu_n) \\ \mu_{n+1} &= \mu_n + \rho(A\beta_{n+1} + B\theta_{n+1} - c).\end{aligned}$$

For the lasso

$$\begin{aligned}\text{Find} &\quad \beta, \theta \\ \text{To minimize} &\quad \frac{1}{2} \|y - X\beta\|_2^2 + \lambda \|\theta\|_1 \\ \text{Such that} &\quad \beta - \theta = 0,\end{aligned}$$

β is updated by a ridge regression, θ by soft thresholding and μ linearly.

Minorization-maximization (MM) minimizes a (possibly non-convex) function f by finding a function $\Psi : \mathbf{R}^m \times \mathbf{R}^n \rightarrow \mathbf{R}$ such that

$$\begin{aligned}\forall \beta, \theta \quad f(\beta) &\leq \Psi(\beta, \theta) \\ \forall \beta \quad f(\beta) &= \Psi(\beta, \beta)\end{aligned}$$

and iterating $\beta_{n+1} = \underset{\beta}{\text{Argmin}} \Psi(\beta, \beta_n)$.

The proximal gradient can be seen as a MM (a Lipschitz gradient gives Ψ); in a different context, so can the expectation maximization (EM) algorithm (Ψ comes from Jensen's inequality).

A function $f(\cdot, \cdot)$ is *biconvex* if $f(\cdot, \beta)$ and $f(\alpha, \cdot)$ are convex (for instance, $f(\alpha, \beta) = (1 - \alpha\beta)^2$); a set $C \subset A \times B \subset \mathbf{R}^n$ is biconvex if the sections $C_{\alpha, \cdot}$, $C_{\cdot, \beta}$ are convex. *Alternate convex search* (ACS) successively optimizes each block; the function values will converge, but the solution need not; if it does, it is only to a *partial optimum*. For instance, the maximal singular vectors and values of a matrix can be computed

as (this is a generalization of the power method)

$$\begin{aligned} f(\alpha, \beta, s) &= \|X - s\alpha\beta'\|_F^2 \\ \alpha_{n+1} &= \frac{X\beta_n}{\|X\beta_n\|_2} \\ \beta_{n+1} &= \frac{X'\alpha_n}{\|X'\alpha_n\|_2}. \end{aligned}$$

When there are millions of variables, one can use screening rules (dual polytope projection (DPP), sequential DPP, global string rule, sequential strong rule) to reduce their number; the less conservative rules make mistakes, but they can be corrected by checking the KKT conditions and adding the variables that violate them.

6. It is possible to design statistical tests for the lasso:

- Bayesian (MCMC) simulations, after choosing a prior (for instance, the number of variables in the model is often assumed uniform over $\llbracket 0, k \rrbracket$, with $k \ll p$) [this is unrelated to the fact that the lasso is the posterior model (MAP estimator) for a Laplace prior];
- Bootstrap (even slower, but it scales better);
- **Covariance test:** the change in $\text{Cov}(\hat{y}, y)$ as a variable enters the model is asymptotically $\text{Exp}(1)$; contrary to stepwise regression, adaptivity and shrinkage (asymptotically) compensate each other;
- Exact (finite sample) tests for adaptive models whose selection event can be written $Ay \leq b$ (this includes lasso and stepwise regression)

One can also debias the lasso estimate to compute confidence intervals.

7. The singular value decomposition (SVD) provides the best rank- r approximation of a matrix Z :

$$UD_rV' = \underset{M : \text{rank } M=r}{\text{Argmax}} \|Z - M\|_F$$

but missing values complicate the situation.

The *iterated SVD* (fill in the missing values, compute the rank r approximation from the SVD, iterate) tends to overfit.

The *soft-thresholding iterated SVD* (fill in the missing values, compute $US_\lambda(D)V'$, iterate; you can also let λ decrease at each step) solves the problem

$$\underset{M}{\text{Argmin}} \frac{1}{2} \|Z - M\|_F^2 + \lambda \|M\|_*$$

where the Frobenius norm is taken over the non-missing entries of Z .

The biconvex problem

$$\underset{A,B}{\text{Argmin}} \|Z - AB'\|_F^2 + \lambda(\|A\|_F^2 + \|B\|_F^2)$$

gives a 2-dimensional family of matrix factorizations, indexed by (r, λ) (r is the number of columns of A and B).

A vector-valued regression $Y = X\Theta + E$ can be estimated with the group lasso, which sets whole rows of

Θ to zero, or with a nuclear norm penalty, to make it low rank.

$$\underset{\Theta}{\text{Argmin}} \|Y - X\Theta\|_F^2 + \lambda \|\Theta\|_*$$

These models (matrix completion, low rank regression) can be written $y_i = \text{tr}(X_i'\Theta) + \varepsilon$ for various choices of the matrices X_i , and estimated as

$$\underset{\Theta}{\text{Argmin}} \frac{1}{N} \sum (y_i - \text{tr}(X_i'\Theta))^2 + \lambda \|\Theta\|_*.$$

The penalized SVD,

$$\underset{U,D,V}{\text{Argmin}} \|Z - UDV'\|_F^2 + \lambda_1 \|U\|_1 + \lambda_2 \|V\|_1$$

gives sparse singular vectors (the rank-1 problem is biconvex and can be solved by alternating soft-thresholding; for the rank- r problem, start again with $Z - udv'$).

Additive matrix decompositions express a matrix Z as a sum, e.g., of low-rank, sparse and small (noise) matrices:

$$\underset{L,S}{\text{Argmin}} \|Z - (L + S)\|_F^2 + \lambda_1 \|L\|_* + \lambda_2 \|S\|_1.$$

For row-wise sparsity (*i.e.*, to model row-wise corruption), replace $\|S\|_1$ with $\sum \|S_i\|_2$.

Factor analysis models the data as $y_i = \mu + \Gamma u_i + \varepsilon_i$, $\varepsilon_i \sim N(0, S)$, which gives $\text{Var } y = \Gamma\Gamma' + S$. If S is scalar, principal component analysis (PCA) recovers Γ ; if not, one can use the low rank + sparse + noise decomposition.

8. Most statistical procedures can be formulated as an optimization problem: adding an ℓ^1 penalty (or both ℓ^1 and ℓ^2 penalties) sparsifies them. It may be necessary to reformulate the problem to get a convex, or biconvex, or less nasty problem. Different reformulations give different results.

8a. The first principal component is the direction in which the variance is maximal

$$\underset{v : \|v\|_2=1}{\text{Argmax}} \text{Var } Xv = \underset{v : \|v\|_2=1}{\text{Argmax}} v'X'Xv;$$

it can be sparsified as

$$\begin{aligned} \text{Find} & \quad u, v \\ \text{To maximize} & \quad u'Xv \\ \text{Such that} & \quad \|u\|_2 = \|v\|_2 = 1, \quad \|v\|_1 \leq t \end{aligned}$$

(solvable with SVD and thresholding) or

$$\begin{aligned} \text{Find} & \quad M \succcurlyeq 0 \\ \text{To maximize} & \quad \text{tr}(X'XM) \\ \text{Such that} & \quad \text{tr } M = 1, \quad \text{tr}(|M|E) \leq t^2. \end{aligned}$$

The first principal component can also be defined as the direction minimizing the reconstruction error, and sparsified as

$$\underset{v, \theta : \|\theta\|_2=1}{\text{Argmin}} \frac{1}{N} \sum \|x_i - uv'\theta\|_2^2 + \lambda_1 \|v\|_1 + \lambda_2 \|v\|_2^2.$$

Auto-encoders also minimize the reconstruction error and are easy to sparsify.

For the next principal component, since orthogonality may conflict with sparsity, one may consider

$$\begin{aligned} \text{Find} & \quad u_k, v_k \\ \text{To maximize} & \quad u_k' X v_k \\ \text{such that} & \quad \|v_k\|_2 \leq 1, \quad \|u_k\|_2 \leq 1 \\ & \quad \|v_k\|_1 \leq c \\ & \quad \forall j \in \llbracket 1, k-1 \rrbracket \quad u_k' u_j = 0. \end{aligned}$$

Classical PCA is consistent when $N \rightarrow \infty$, $p/N \rightarrow 0$, but not (at all) if $N, p \rightarrow \infty$, $p/N \rightarrow c > 0$. Thresholding the diagonal of the sample variance matrix and computing the PCA in this lower-dimensional space is actually consistent.

8b. Canonical correlation analysis (CCA) solves the problem

$$\begin{aligned} \text{Find} & \quad \beta, \theta \\ \text{To maximize} & \quad \text{Cov}(X\beta, Y\theta) \\ \text{Such that} & \quad \text{Var } X\beta = \text{Var } Y\theta = 1 \end{aligned}$$

e.g., via SVD or ALS; it can be sparsified by adding ℓ^1 and ℓ^2 constraints and solved by alternating soft-thresholding.

8c. Linear discriminant analysis (LDA) can be presented in three different ways, which can be sparsified as usual:

- Model the classes with Gaussian distributions, with different means but the same variance (the naive Bayes classifier is the special case when the variance matrix is diagonal);
- Fisher: find a low-dimensional projection in which the between-class variance is large wrt the within-class variance, *i.e.*,

$$\text{Argmax}_{\beta} \frac{\beta' \Sigma_b \beta}{\beta' \Sigma_w \beta};$$

- Optimal scoring: LDA with two classes is a linear regression of the binary response; with more classes, it is still a linear regression, provided we assign a number to each class, in an optimal way.

8d. Hierarchical clustering does not work well in the presence of many uninformative variables: *sparse hierarchical clustering* assigns (sparse) weights to the variables and applies the usual clustering algorithms on the resulting weighted dissimilarity matrix.

Convex clustering looks for prototypes, close to the points, and close to one another (with an ℓ^1 loss, so that many are actually equal),

$$\text{Argmin}_{u_1, \dots, u_n} \sum \|x_1 - u_i\|_2^2 + \lambda \sum_{i < j} \|u_i - u_j\|_q, \quad q \in \{1, 2\}.$$

K -means can be sparsified by maximizing the between-

cluster sum of squares

$$\begin{aligned} \text{Find} & \quad \mathcal{C}_1 \sqcup \dots \sqcup \mathcal{C}_K = \llbracket 1, N \rrbracket, \quad w \in \mathbf{R}^p \\ \text{To maximize} & \quad \sum_{j=1}^p w_j \left(\frac{1}{N} \sum_{i,i'=1}^N d_{ii'j} - \sum_k \frac{1}{N_k} \sum_{i,i' \in \mathcal{C}_k} d_{ii'j} \right) \\ \text{Such that} & \quad \|w\|_2 \geq 1, \quad \|w\|_1 \leq s, \quad w \geq 0 \end{aligned}$$

and solved with an alternating algorithm (soft-thresholding and weighted k -means).

9. Undirected graphs can encode dependence properties between random variables (one per vertex) in the following equivalent ways (*Hammersley-Clifford theorem*):

- The joint probability distribution function factorizes over the cliques

$$P(x_1, \dots, x_p) \propto \prod_{C \in \text{Cliques}} \psi_C(x_C);$$

- The random variable X has the Markov property over the graph: for all cut set S separating the graph into connected components A and B , $X_A \perp\!\!\!\perp X_B | X_S$.

For instance, one can model how politicians vote as follows

$$\begin{aligned} X_i &: \text{vote of politician } i \\ \text{sign } \theta_i &: \text{whether } i \text{ is likely to vote yes} \\ \text{sign } \theta_{ij} &: \text{whether } i \text{ and } j \text{ tend to agree} \\ P &\propto \exp\left(\sum \theta_i x_i + \sum \theta_{ij} x_i x_j\right) \end{aligned}$$

(the graph is complete, but only cliques up to size 2 are used).

A multivariate Gaussian model is a graphical model, with only vertex and edge factors; the adjacency matrix of the graph is the sparsity pattern of the concentration (or precision) matrix $\Theta = \Sigma^{-1}$.

The Gaussian likelihood can be penalized (**graphical lasso**)

$$\text{Argmax}_{\Theta \succeq 0} \log \det \Theta - \text{tr } \hat{\Sigma} \Theta - \lambda \rho_1(\Theta)$$

where ρ_1 is the ℓ^1 norm of the off-diagonal elements and $\lambda = 2\sqrt{\frac{\log p}{N}}$. The problem is convex and can be solved efficiently by blockwise coordinate descent (fix everything except a row (and the corresponding column); each step is a lasso regression). If the solution has a block structure, it can be identified before the optimization ($|\hat{\Theta}_{ij}| \leq \lambda$ whenever i and j are in different blocks) and the blocks can be solved independently.

Alternatively, one can estimate the neighbourhood of a random variable as a small set of variables that gives a good prediction, using the lasso (and the AND or OR rule: keep an edge (s, t) if both/either $s \in \mathcal{N}(t)$ and/or $t \in \mathcal{N}(s)$). For discrete variables, use the logistic lasso.

It is possible to mix continuous and discrete variables

$$\begin{aligned} \log P &= \sum \gamma_s x_s - \frac{1}{2} \sum \theta_{st} x_s x_t && \text{continuous} \\ &+ \sum \rho_{sj}(y_j) x_s && \text{mixed} \\ &+ \sum \psi_{jr}(y_j, y_r) && \text{discrete} \end{aligned}$$

through the *pseudo-likelihood* (the product of the neighbourhood likelihoods).

Omitting hidden variables makes a sparse model look dense. In this case, the concentration matrix can be written as a sum $K = \Theta + L$, with Θ sparse and L low rank (from the block inverse formula):

$$\text{Argmin}_{\substack{\Theta, L \\ \Theta - L \succ 0 \\ L \succ 0}} \text{tr} \hat{\Sigma}(\Theta - L) - \log \det(\Theta - L) + \lambda \|\Theta\|_1 + \text{tr} L.$$

10. The best sparse approximation, in an orthonormal basis (Fourier, wavelets) is obtained by keeping the k largest coefficients. One can also combine several bases, e.g., Fourier Φ to capture periodic patterns and smooth changes, and Haar Ψ for sharp changes, with a lasso constraint

$$\begin{aligned} \text{Find} & \quad \alpha, \beta \\ \text{To minimize} & \quad \|\theta - \Phi\alpha - \Psi\beta\|_2 \\ \text{Such that} & \quad \|\alpha\|_1 + \|\beta\|_1 \leq R. \end{aligned}$$

The exact reconstruction problem is even simpler:

$$\begin{aligned} \text{Find} & \quad \alpha, \beta \\ \text{To minimize} & \quad \|\alpha\|_1 + \|\beta\|_1 \\ \text{Such that} & \quad \theta = \Phi\alpha - \Psi\beta, \end{aligned}$$

Compressed sensing reconstructs a sparse signal θ (e.g., the signal of interest, in a basis in which we expect it to be sparse) from a small number of random projections $z'_i \theta$. The random projection matrix can have Gaussian iid entries, Bernoulli ± 1 iid entries, or be a random submatrix of the Fourier matrix $(\exp(2\pi i k \ell / n))_{1 \leq k, \ell \leq n}$.

In R, check the following packages: `glmnet` (GLM), `genlasso` (fused lasso), `monomvn::blasso` (Bayesian lasso, bootstrap), `softImpute` (SVD with missing values), `penalizedLDA`, `sparseLDA`, `cvxcluster` (convex clustering).

Selected applications of convex optimization L. Li (2015)

1. Given the primal problem

$$\begin{aligned} \text{Find} & \quad x \\ \text{To minimize} & \quad f(x) \\ \text{Such that} & \quad \forall i \ g_i(x) \leq 0 \\ & \quad \forall j \ h_j(x) = 0, \end{aligned}$$

define the Lagrangian and the dual function as

$$\begin{aligned} L(x, \alpha, \beta) &= f(x) + \sum \alpha_i g_i(x) + \sum \beta_j h_j(x) \\ q(\alpha, \beta) &= \inf_x L(x, \alpha, \beta); \end{aligned}$$

the dual problem is then

$$\begin{aligned} \text{Find} & \quad \alpha, \beta \\ \text{To maximize} & \quad q(\alpha, \beta) \\ \text{Such that} & \quad \alpha \geq 0. \end{aligned}$$

2. A support vector machine (SVM) can be fit by

$$\begin{aligned} \text{Find} & \quad w, b \\ \text{To maximize} & \quad 1/\|w\|_2^2 \quad (\text{width of the band}) \\ \text{Such that} & \quad \forall i \in L^+ \ w'x_i + b \geq 1 \\ & \quad \forall i \in L^- \ w'x_i + b \leq -1, \end{aligned}$$

which can be written as a convex problem

$$\begin{aligned} \text{Find} & \quad w, b \\ \text{To minimize} & \quad \|w\|_2^2 \\ \text{Such that} & \quad \forall i \ y_i(w'x_i + b) \geq 1 \end{aligned}$$

where $y_i = +1$ if $i \in L^+$ and -1 otherwise. To find the dual problem, write the Lagrangian

$$L(z, b, \alpha) = \frac{1}{2} \|w\|_2^2 - \sum \alpha_i (y_i(w'x_i + b) - 1);$$

use the KKT conditions, $\partial L / \partial w = 0$ and $\partial L / \partial b = 0$, i.e., $w = \sum \alpha_i y_i x_i$ and $\sum \alpha_i y_i = 0$ to compute the dual function $q(\alpha)$:

$$\begin{aligned} \text{Find} & \quad \alpha \\ \text{To maximize} & \quad \sum \alpha_i - \frac{1}{2} \sum \alpha_i \alpha_j y_i y_j x'_i x_j \\ \text{Such that} & \quad \alpha \geq 0 \\ & \quad \sum \alpha_i y_i = 0. \end{aligned}$$

There are many variants, such as the soft-margin SVM

$$\begin{aligned} \text{Find} & \quad w, b, \xi \\ \text{To minimize} & \quad \|w\|_2^2 + c \sum \xi_i^2 \\ \text{Such that} & \quad \forall i \ y_i(w'x_i + b) \geq 1 - \xi_i \\ & \quad \forall i \ \xi_i \geq 0 \end{aligned}$$

or the multiclass SVM (with $t_i = [x_i, 1]$)

$$\begin{aligned} \text{Find} & \quad v_1, \dots, v_k, \xi \\ \text{To minimize} & \quad \frac{1}{2} \sum v'_k v_k + c \sum \xi_i^2 \\ \text{Such that} & \quad \forall i, k \ v'_i t_i - v'_k t_i \geq \delta_{y_i, k} - \xi_i. \end{aligned}$$

Sequential minimal optimization (SMO) tries to reduce memory usage by optimizing two α_i 's at a time.

One can also regularize SVMs, by replacing $\sum \xi_i$ with $\sum \xi_i^2$ (LS-SVM), or by a constraint such as $\sum \xi_i^2 = L\sigma^2$ (Morozov) or $w'w = \pi^2$ (Ivanov).

3. A mixture model

$$\begin{aligned} f(x, \theta) &= \sum \pi_k f(x, \theta_k) \\ z_i(k) &= \mathbf{1}_{x_i \text{ comes from the } k\text{th component}} \end{aligned}$$

can be estimated by expectation maximization (EM):

- E-step: estimate the distribution (not the values – that is a common mistake) of $z_i(k)$ from (π, θ) :

$$\begin{aligned}\gamma_i(k) &= P[x_i \text{ comes from the } k\text{th component}] \\ &= E[z_i(k)] \\ &= \frac{\pi_k f(x_i, \theta_k)}{\sum_j \pi_j f(x_i, \theta_j)}\end{aligned}$$

- M-step: estimate (π, θ) from γ via maximum likelihood.

The general EM algorithm estimates a parameter of interest θ and integrates out nuisance parameters (latent variables or missing values) z , by estimating their posterior distribution $q_i(z_i)$. The log-likelihood can be written

$$\begin{aligned}\log L(\theta) &= \sum_i \log p(x_i, \theta) \\ &= \sum_i \log \sum_{z_i} p(x_i, z_i, \theta) \\ &= \sum_i \log \sum_{z_i} q_i(z_i) \frac{p(x_i, z_i, \theta)}{q_i(z_i)} \\ &\geq \sum_i \sum_{z_i} q_i(z_i) \log \frac{p(x_i, z_i, \theta)}{q_i(z_i)} \quad (\text{Jensen}) \\ &= J(q, \theta)\end{aligned}$$

and maximized by coordinate descent:

- E-step: $q_m = \text{Argmax}_q J(q, \theta_m)$ (we do not estimate the nuisance parameters z_i but their distributions q_i ; after this step, the inequality is actually an equality)
- M-step: $\theta_{m+1} = \text{Argmax}_\theta J(q_m, \theta)$.

The book then gives probabilistic principal component analysis (PCA with missing values) as an example, but they make the common mistake of filling in the missing values.

4. Adding a penalty to the objective, e.g., $\lambda \|x\|_1$, is equivalent to adding a constraint $\|x\|_1 \leq C$, where $C = \|x^*\|_1$ and x^* is the solution of the penalized problem.

6. The SDP (semidefinite programming) relaxation of a non-convex quadratically constrained quadratic program (QCQP, e.g., $x \in \{\pm 1\}$ can be written $x^2 = 1$) is obtained by introducing a matrix $X = xx'$ and relaxing that equality to $X \succcurlyeq xx'$, i.e.,

$$\begin{pmatrix} 1 & x' \\ x & X \end{pmatrix} \succcurlyeq 0.$$

The *reformulation-linearization technique* (RLT) also relaxes $X = xx'$, but start with the inequalities $\ell_i \leq x_i \leq u_i$, i.e., $x_i - \ell_i \geq 0$ and $u_i - x_i \geq 0$, multiplies them, and replaces $x_i x_j$ with x_{ij} .

7. The minimum volume ellipsoid covering n points $x_1, \dots, x_n \in \mathbf{R}^d$ is defined by

$$\begin{array}{ll}\text{Find} & A, \mu \\ \text{To minimize} & \det Q \\ \text{Such that} & \forall i \ (x_i - \mu)' Q^{-1} (x_i - \mu) \leq 1 \\ & Q \succ 0\end{array}$$

and can be recast as a convex problem

$$\begin{array}{ll}\text{Find} & M, m \\ \text{To minimize} & -\log \det M \\ \text{Such that} & \forall i \ (Mx_i - m)'(Mx_i - m) \leq 1 \\ & M \succ 0\end{array}$$

where $M = Q^{-1/2}$ and $m = Q^{-1/2}\mu$.

Parsing algorithms

Recursive descent is the most straightforward top-down parsing algorithm: start with the root of the tree, try the first possible derivation, and continue, backtracking after each failure (it does not work with left recursive (LR) grammars, i.e., those with a rule of the form $S \rightarrow^+ S\alpha$, where \rightarrow^+ means a series of derivations). *Parser combinators* (Parsec, Spirit) are a way to implement recursive descent parsing in functional languages, from simple building blocks (parsers, i.e., functions that convert strings to parse trees) and glue – higher order functions).

Parsing expression grammars (PEG) are an alternative to context-free grammars (CFG): the choice operator $|$ is ordered; the $*$ operator is greedy and does not backtrack after failure; there are $\&$ (AND) and $!$ (NOT) operators (they do not consume any input and allow unlimited look-ahead). *Packrat* is a linear-time recursive descent algorithm with memoization for PEG.

The **Earley** algorithm is similar to recursive descent, but explores the set of possible parse trees breadth-first, not depth-first.

The **CYK** algorithm is a bottom-up dynamic programming based parsing algorithm, building a boolean table in $O(n^3)$: T_{ijk} indicates if $s_{i..j}$ can be derived from rule R_k (it assumes the grammar is in Chomsky normal form (CNF) i.e., the derivation rules are of the form $A \rightarrow \alpha$ or $A \rightarrow BC$).

A grammar is $LL(k)$ if, whenever we hesitate between two derivation rules, we can decide by looking k tokens ahead. An **LL** parser (e.g., ANTLR) precomputes those decisions and puts them in a table (the table is exponential in the worst case, but not in practice).

LR parsers are a bottom-up analogue of the top-down LL parsers: they use a transition table that maps parser state and next token(s) to a next state and an action; the action can be “shift” (consume one token) or “reduce” (do not, but apply one of the rules). Variants include LALR(1) (look-ahead LR(0), yacc/bison), SLR (simple LR), GLR (generalized LR).

QuantifQuantile: an R package for performing quantile regression through optimal quantization
I. Charlier et al. (2015)

Nonparametric quantile regression $Y \sim X$ can be estimated with a variant of the k nearest neighbours:

- Find a set (or grid) γ of N points to minimize $E[|X - X^\gamma|^p]^{1/p}$, where X^γ is the projection of X onto γ (start with a random grid γ and move each grid point towards the average of the observations assigned to it);
- Approximate the conditional quantile of $Y|X$ as

$$\hat{q}_\alpha(y|x) = \underset{a \in \mathbf{R}}{\text{Argmin}} E[\rho_\alpha(Y - a) | X^\gamma = x^\gamma]$$

- Bootstrap to choose N ;
- Bootstrap for a smoother estimate.

***treeClust: an R package
for tree-based clustering dissimilarities***
S.E. Buttrey and L.R. Whitaker

Tree-based distances are insensitive to scaling and provide automatic variable selection: use the proportion of trees (predict each variable with the others) in which the two observations are in the same leaf; refine using tree quality and node distance.

***An R package for the panel approach method
for program evaluation: pampe***
A. Vega-Bayo

To estimate the effect of a policy (“program evaluation”) from observed panel data, with treated and control groups, consider the effect of treatment/non-treatment on the control/treated group as missing values, and model the panel data with a dynamic factor model.

A significance test for the lasso
R. Lockhart et al. (2014)

To test if the inclusion of a variable along the regularization path is significant, use

$$T_k = \frac{\langle y, X\hat{\beta}(\lambda_{k+1}) - X_{A_{k-1}}\tilde{\beta}_{A_{k-1}}(\lambda_{k+1}) \rangle}{\sigma^2} \xrightarrow{d} \text{Exp}(1)$$

where $\hat{\beta}(\lambda)$ is the lasso estimator, A_{k-1} is the set of variables in the model just before λ_k , $\tilde{\beta}_A(\lambda)$ is the lasso estimator on the variables in A .

***Automatically improving accuracy
for floating point expressions***
P. Panchekha (2015)

Herbie is a Racket library that rewrites floating point expressions to improve their precision (e.g., $e^x - e^{-x}$ for x small, or $\log(1 + e^x)$ for x large):

- Sample the possible inputs, uniformly among floating point numbers (not uniformly over some interval of \mathbf{R});
- Estimate the precision of each intermediate result, by comparing it with an arbitrary precision computation (progressively increase the precision until it is sufficient);
- Apply (more than a hundred) rewrite rules (e.g., associativity, but also $\sqrt{x+1} - \sqrt{x} \mapsto 1/(\sqrt{1+x} + \sqrt{x})$); simplify;

- Also try series expansions;
- Choose which rewritten formula to use depending on the values of the inputs.

TOL manual
(2013)

TOL (time-oriented language) is a programming language with predefined types to describe time series:

- Date and Time (confusingly, it is the same type);
- Infinite sets of dates (e.g., the first day of each month), with operations on them (union, intersection, set difference, membership, next-element, restriction, offset);
- Unlimited time series, with a few building blocks (indicator, step function, affine function, random number generator) and operations (+, −, ×, /, coalesce, lag, arima, etc.).

The package repository has a few dozen items, but many are only documented in Spanish.

It is not clear why this DSL was not built on an existing language (Haskell, OCaml, etc.).

Mathematica: a problem-centered approach
R. Hazrat (2010)

Here are a few examples of the syntax of the language.

- Square brackets are used for function arguments, curly braces for lists; matrices are lists of lists;
- Function definitions use `:=`, while `=` creates frozen assignments: compare `x:=Random[]` and `x=Random[]`;
- Function arguments are indicated by `_`, and they can be given an optional name, e.g., `f[n_]:=n^2`;
- Function application can be written `f[x]`, `f@x` or `x//f`; the pipe operator is often used to format the output, e.g., `Inverse[A] // MatrixForm`;
- Anonymous functions end with `&` (it is not always clear where they start) and their arguments are called `#1`, `#2`, etc. (or `#` of there is only one), e.g., `(#^2+1)&[5]` or `5/(#^2)&`;
- The map function is `/@`, e.g., `f /@ Range[5]`; the function is often anonymous, e.g., `#^2& /@ Range[5]`;
- The fold function is `Apply` or `@@`, e.g., `Plus @@ Range[5]` (it actually replaces the head of the rhs with the lhs); its vectorized form, for lists of lists, is `@@@`;
- The `Table` function provides a weak form of list comprehension, e.g., `Table[3n^5+11, {n,1,20}]`;
- The substitution operator is `./`, e.g., `x+y ./ x->2`; `./` applies the rules until nothing changes, e.g., `x+y ./ {x->y,y->z}`;
- It is possible to specify the type and properties of the function arguments or substitution, e.g., `f[x_Integer?EvenQ]:=x/2`; `__` matches a non-empty sequence, `___` a sequence;
- If can be written `/;`, e.g., `f[n_]:=Sqrt[n] /; n>0`.

***A thread-safe arbitrary precision
computation package***
D.H. Bailey (2015)

Implementation details for a multiprecision library – useful for ODEs (e.g., the Lorentz attractor loses 16 digits in each period) and experimental mathematics. Alternatives include: GMP, MPFR (and languages that use it, e.g., Julia), Pari/GP, mpmath (Python), Sage.

Use Newton iteration for square roots $x \leftarrow x + \frac{1}{2}(1 - x^2a)x$, n th roots, inverses $x \leftarrow x + (1 + xb)x$, logarithm $x \leftarrow x - (e^x - a)/e^x$, arcsine; Taylor for exponential (for x small), sine (for x small – use the double angle formula to bring it back to its initial value in $(-\pi/4, \pi/4]$). For multiplication, use the elementary school algorithm (in base 2^{48}) or the FFT (the digits of ab , before accounting for carries, are the convolution of those of a and b).

For much higher precision (thousands to millions of digits), algorithms based on the *arithmetic-geometric mean* (AGM), being quadratically convergent (the number of correct digits doubles at each iteration), become competitive.

Ten problems in experimental mathematics
D.H. Bailey et al. (2006)

Applications of the PSLQ algorithms to situations where the numeric estimation of the number of interest is non-trivial.

***Using integer relation algorithms
for finding relationships among functions***
M. Chamberland (2007)

The PSLQ algorithm can also find integer relations between functions: evaluate them at some random point; apply PSLQ; try another point to make sure.

A=B
M. Petkovšek et al. (1997)

A geometric series is a series $\sum t_n$ in which the ratio of two consecutive terms t_{n+1}/t_n is constant. A **hypergeometric series** is a series $\sum t_n$ in which the ratio of two consecutive terms t_{n+1}/t_n is a rational function of n . If

$$\frac{t_{n+1}}{t_n} = \frac{(n+a_1) \cdots (n+a_p)}{(n+b_1) \cdots (n+b_q)} \frac{x}{n+1},$$

it is usually written

$${}_pF_q \left(\begin{smallmatrix} a_1 \cdots a_p \\ b_1 \cdots b_q \end{smallmatrix}; x \right) = {}_pF_q(\mathbf{a}, \mathbf{b}; x) = \sum_{n \geq 0} t_n.$$

There are lists of hypergeometric identities, but expecting to be able to evaluate arbitrary hypergeometric series from those lists is hopeless.

1. Given a double hypergeometric term $F(n, k)$, *i.e.*, both $F(n+1, k)/F(n, k)$ and $F(n, k+1)/F(n, k)$ are

rational functions, **Sister Celine’s algorithm** looks for a recurrence relation

$$\sum_{i=0}^I \sum_{j=0}^J a_{ij}(n) F(n-j, k-i) = 0$$

as follows: divide the recurrence relation by $F(n, k)$; simplify; only rational functions remain: put them on the same denominator; the numerator is a polynomial in k and its coefficients should be zero: solve the corresponding system; if it does not have any non-trivial solution, increase I and J and try again. By summing over k , we get a recurrence relation for $f(n) = \sum_k F(n, k)$; solve it with the Hyper algorithm.

(The algorithm does not scale well: prefer the equivalent Zeilberger algorithm.)

2. Given a hypergeometric term $(t_k)_k$, *i.e.*, $r_k = t_{k+1}/t_k$ is a rational function of k , **Gosper’s algorithm** looks for another hypergeometric term $(z_k)_k$ such that $\Delta z = t$, *i.e.*, $z_{n+1} - z_n = t_n$; this gives a simple formula for the partial sums $s_n = \sum_{k=0}^{n-1} t_k = z_n - z_0$.

Since it computes an antiderivative of a hypergeometric series, it can be seen as an analogue of the Risch algorithm (differential Galois theory) for hypergeometric series.

The details are technical: write

$$r + n = \frac{a_n}{b_n} \frac{c_n + 1}{c_n}$$

with a, b, c polynomials and $\forall k \gcd(a_n, b_{n+k}) = 1$; find a polynomial x_n , if it exists, such that $a_n x_{n+1} - b_{n-1} x_n = c_n$; let

$$z_n = \frac{b_{n-1} x_n}{c_n} t_n.$$

3. Zeilberger’s algorithm (aka creative telescoping) is a (faster) variant of Sister Celine’s algorithm, to find a telescoping recurrence relation

$$F(n+1, k) - F(n, k) = G(n, k+1) - G(n, k)$$

or, more generally,

$$\sum_{j=0}^J a_j(n) F(n+j, k) = G(n, k+1) - G(n, k).$$

It is guaranteed to exist if F is a proper hypergeometric term:

$$F(n, k) = P(n, k) \frac{\prod_{i=1}^U (a_i n + b_i k + c_i)!}{\prod_{j=1}^V (u_j n + v_j k + w_j)!} x^k,$$

where P is a polynomial, U, V, a_i, b_i, u_j, v_j are integers (c_i and w_j are arbitrary).

Summing over k gives a recurrence relation for $s_n = \sum_k F(n, k)$ (the rhs cancels out if G has finite support in k for all n).

4. To prove hypergeometric identities, e.g., $\sum_k F(n, k) = 1$, the **Wilf-Zeilberger (WZ) algorithm** finds G (with compact support in k for all n) such that $F(n-1, k) - F(n, k) = G(n, k+1) - G(n, k)$, e.g., by applying Gosper's algorithm.

5. The **Hyper algorithm** finds the closed form solution, if it exists, of any recurrence with polynomial coefficients (*holonomic sequence*) by looking for a hypergeometric solution and reducing the problem to finding polynomial solutions of another recurrence (here, "closed form" means sum of hypergeometric series).

Note that the zeroes of the coefficients of the recurrence can make the dimension of the space of solutions larger than expected: checking that two solutions of a recurrence of degree k agree on just k points may not be sufficient to prove they are equal.

There are implementations of those algorithms in Maple (EKHAD) and Mathematica (Gosper, Hyper, WZ).

Deep reinforcement learning with double Q-learning H. van Hasselt et al. (2016)

Imprecise action values lead to overestimated values. Double Q learning, which learns two sets of weights, θ and θ' , randomly updating one of them at each step, and replaces the Q -learning target

$$\begin{aligned} Y_t^Q &= R_{t+1} + \gamma \text{Max}_a Q(S_{t+1}, a; \theta_t) \\ &= R_{t+1} + \gamma Q(S_{t+1}, \text{Argmax}_a Q(S_{t+1}, a, \theta_t), \theta_t) \end{aligned}$$

with $Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \text{Argmax}_a Q(S_{t+1}, a, \theta_t), \theta'_t)$,

reduces the problem.

Tensorizing neural networks A. Novikov et al.

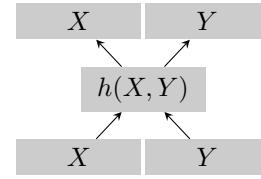
To reduce the memory footprint of large neural nets, one can use lower numeric precision, hashing, or low rank tensor approximations such as Tucker, CP or tensor train (TT). A TT decomposition of a tensor \mathcal{A} is a decomposition of each element as a product of matrices (the first and last factors are row and column matrices)

$$\mathcal{A}(j_1, \dots, j_d) = G_{1j_1} G_{2j_2} \cdots G_{dj_d}.$$

Correlation neural networks S. Chandar et al.

Canonical correlation analysis (CCA – for a C++ implementation, check `dlib`) solves the following problem: given two (vector) random variables X, Y , find x and y to maximize $\text{Cor}(x'X, y'Y)$; $x'X$ and $y'Y$ can be seen as a common representation of X and Y . An

auto-encoder can be used for the same purpose,



minimizing the sum of

- Reconstruction error of (X, Y) given (X, Y) ;
- Reconstruction error of (X, Y) given $(X, 0)$;
- Reconstruction error of (X, Y) given $(0, Y)$;
- Minus $\text{Cor}(h(X, 0), h(0, Y))$.

[One could also use a noisy auto-encoder: during training, replace either X or Y with noise.]

Semi-supervised learning with ladder networks A. Rasmus et al.

A *ladder network* is an auto-encoder with skip connections, to preserve details.

Scheduled sampling for sequence prediction with recurrence neural networks S. Bengio et al.

Neural networks outputting a sequence usually predict the next token from the current (hidden) state and the previous token – there may be a discrepancy between training and inference, the former using the true token while the latter only has the precious predicted token. Instead, during training, one can randomly choose to use the true or predicted token, with the probability of the true token decreasing as training progresses.

Learning both weights and connections for efficient neural networks S. Han et al.

For sparse neural nets: train; prune unimportant connections; iterate.

Minimum HGR correlation principle: from marginals to joint distribution F. Farnia et al.

The HGR correlation is

$$\begin{aligned} \rho(X, Y) &= \sup \{ E[f(X)g(Y)] : f, g \text{ measurable such} \\ &\quad \text{that } EfX = EgY = 0 \text{ and} \\ &\quad E(fX)^2 = E(gY)^2 = 1 \}. \end{aligned}$$

It is between 0 and 1, and is 0 (resp. 1) iff X and Y are independent (strictly dependent). The distribution with the minimum HGR correlation for given first and second moments is the Gaussian – this is an analogue of the maximum entropy property of the Gaussian distribution.

**Scalable Bayesian optimization
using deep neural networks**
J. Snoek et al.(2015)

Bayesian neural networks, *i.e.*, distributions over the weights of a neural net (instead of point estimates) are computationally expensive. They can be approximated by *adaptive basis regression*, *i.e.*, neural nets in which only the last, linear layer is Bayesian. Bayesian optimization with Gaussian processes scales cubically in the number of observations; adaptive basis regression is a linear alternative (random forests are another).

**Gradient-based hyperparameter optimization
through reversible learning**
D. Mcclaurin et al.

Backpropagation through the entire learning procedure, back to the hyperparameters, allows the use of gradient-based (instead of gradient-free Bayesian optimization) hyperparameter tuning. Storing all the intermediate results needed to compute the gradients is unreasonable (it would mean remembering all the iterations of the inner loop, on the mini-batches). It is however possible to run the optimization procedure backwards, provided we remember all the bits that fall off in floating point computations.

**Topological data analysis of contagion maps
for examining spreading processes on networks**
D. Taylor et al.

The Watts threshold model (WTM) map is obtained by simulating the WTM (a node becomes infected if the proportion of infected neighbours exceeds some threshold) and using the infection times (for various (singleton) infection seeds) as coordinates; dimension reduction may be needed.

**Compressing neural networks
with the hashing trick**
W. Chen et al.

To reduce the size of a neural net:

- Train a 1-layer net on the log-output of a deep net;
- Do not learn the full weight matrices but low rank approximations;
- Use feature hashing to reduce the weight matrices to a reasonable number of parameters.

**Analyzing big data
with dynamic quantum clustering**
M. Weinstein et al.

The *mean-shift* clustering algorithm starts with a cloud of points, computes a density estimator, moves each point uphill, and iterates. Dynamic quantum clustering (DQC) is a quantum-flavoured variant:

- Start with a cloud of points x_1, \dots, x_n ;
- Consider the density estimator $\phi(x) = \sum \phi(x_i)$,
 $\phi(x_i) = \exp -\frac{1}{2}(x - x_i)'(x - x_i)/\sigma^2$;

- Consider the potential V defined by $H\phi = 0$, where the Hamiltonian operator is

$$H = -\frac{1}{2\sigma^2}\nabla^2 + V(x);$$

- Move the densities towards this potential $\phi_1 \mapsto e^{-i\delta H}\phi_1$;
- Compute the new position of the points;
- Iterate.

**Input warping for Bayesian optimization
of non-stationary functions**
J. Snoek et al.

Gaussian processes (GP) usually model stationary processes: the covariance function $k(x, y)$ only depends on $x - y$. For non-stationary processes, one can use a hierarchical Bayesian model, including a transformation of the input space to make the process stationary. For instance, the cumulative distribution function of the beta distribution can model simple monotonic transformations: use a logarithmic prior on α and β to encode the prior belief in an exponential, logarithmic, linear, sigmoid or logit transformation.

**A sufficient and necessary condition
for global optimization**
D.H. Wu et al. (2010)

If f is Lebesgue-integrable, then

$$\text{essinf } f = \sup\{\alpha : V_{m,f}(\alpha) = 0\},$$

$$V_{m,f}(\alpha) = \int (\alpha - f(x))_+^m d\mu$$

$$V_{m,f}^{(m)}(\alpha) = m!\mu[f(x) \leq \alpha].$$

Use Monte Carlo simulations to estimate those integrals and turn the formula into a global optimization algorithm.

**Chemical equation balancing:
an integer programming approach**
S.K. Sen et al. (2006)

Balancing a chemical equation is a linear optimization problem:

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & x' \mathbf{1} \\ \text{Such that} & Ax = 0, \ x \geq \mathbf{1}, \ x \in \mathbf{Z}^n. \end{array}$$

The quadratic problem

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & x'x \\ \text{Such that} & Ax = 0, \ x \geq \mathbf{1}, \ x \in \mathbf{Z}^n \end{array}$$

may be easier to solve and often has the same solution.

***A new approach
to balancing chemical equations***
I.C. Risteski

Balancing a chemical equation means finding positive vectors, x and y , if they exist, so that $Ax = By$, where A and B have nonnegative entries. Try:

$$\begin{aligned}y &= (I - G^+G)u \\x &= A^+By + (I - A^+A)v \\G &= (I - AA^+)B\end{aligned}$$

for arbitrary vectors u and v . (You need to compute a pseudo inverse in \mathbf{Q} , exactly, or in $\mathbf{Z}/n\mathbf{Z}$, with n sufficiently large.)

***Mastering the game of Go
with deep neural networks and tree search***
D. Silver et al. (2016)

Consider a Go board as a 19×19 image and feed it to two deep convolutional neural nets, for forecast the next move (supervised learning from human games), and to estimate the value of a position; combine with Monte Carlo tree search (MCTS).

Ball arithmetic
J. van der Hoeven (2011)

For reliable computing, consider replacing interval arithmetic with ball arithmetic (sometimes called midpoint-radius arithmetic):

- While interval arithmetic require full precision for both ends of the intervals, ball arithmetic only needs full precision for the centers;
- On \mathbf{C} or $\mathbf{R}^{n \times m}$, intervals tend to grow faster than balls (e.g., $z \mapsto z^2$ around $z = 1 + i$).

Mathemagix if a (C++) software environment for certified numeric computations.

Arb: a C library for ball arithmetic
F. Johansson

Accurate special functions.

Bitcoin and cryptocurrency technologies
A. Narayanan et al. (Princeton & Coursera, 2015)

Non-technical but clear and comprehensive exposition of cryptocurrencies and their applications.

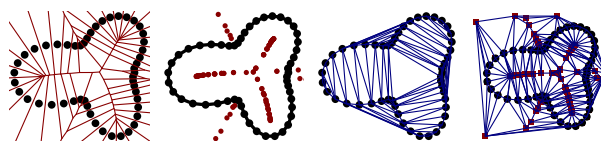
***Curve and surface reconstruction algorithms
with mathematical analysis***
T.K. Dey (2007)

The **skeleton** of a shape (curve, surface) $\Sigma \subset \mathbf{R}^k$ is the set of centers of balls that do not contain any point of Σ and cannot be enlarged while keeping this property. The **medial axis** M_Σ is its closure. It is also the closure of the set of points that have at least two closest points in Σ . [Those notions also

appear in *mathematical morphology*, a branch of image analysis.] The **local feature size** of $x \in \Sigma$ is $f(x) = d(x, M_\Sigma)$. A sample $P \subset \Sigma$ is an ε -sample if $\forall x \in \Sigma \exists p \in P \|x - p\| \leq \varepsilon f(x)$; if $\varepsilon < 1$, the sample is *dense*.

Given an ε -sample P of a closed plane curve Σ whose local feature size is always positive, one can find a piecewise approximation of Σ , as a subset of the Delaunay triangulation, as follows:

- Compute the Voronoi tessellation of P and the voronoi vertices V ; they approximate the median axis;
- Compute the Delaunay triangulation of $P \cup V$; keep the edges connecting two points of P .



Alternatively, link each point $p \in P$ to its nearest neighbour q and its half-neighbour s – the closest point s such that $\angle spq \geq \pi/2$. Both algorithms are $O(n \log n)$ and work if $\varepsilon < 1/5$, resp. $\varepsilon < 1/3$.

One can reconstruct a compact surface (without boundary) $\Sigma \subset \mathbf{R}^3$ from a sample P of points and its Voronoi tessellation:

- The normal at a point $p \in P$ can be approximated by the direction in which the Voronoi cell V_p is elongated; more formally, if V_p is unbounded, use the average direction v_p of the unbounded edges; if it is bounded, take the point $p^+ \in P$ farthest from p , then the point p^- farther from p among the points such that $\angle p^-pp^+ > \pi/2$ and set $v_p = p^- - p$;
- The surface in the neighbourhood of a point $p \in P$ can be approximated by the *cocone*: the complement of the double cone, centered on p , in the direction v_p , of angle $3\pi/8$, restricted to V_p ;
- Take the Voronoi edges that intersect the cocones; their duals are triangles in the Voronoi tessellation (but there are still too many);
- Remove triangles adjacent to “sharp” edges (angle $< 3\pi/2$);
- “Pockets” (blisters) remain on the approximate surface: remove them by walking on the outside.

***Topological anomaly detection performance
with multispectral polarimetric imagery***
M.G. Gartley and W. Basener (2009)

Outliers form the small connected components of the truncated graph.

Return to the Riemann integral
R.G. Bartle (1996)

A *tagged partition* of an interval $[a, b]$ is the datum of

$$a = x_0 < x_1 < \cdots < x_n = b$$

and, for each i , $t_i \in (x_{i-1}, x_i)$. The *Riemann sum* of a function $f : [a, b] \rightarrow \mathbf{R}$ on a tagged partition $((x_i)_i, (t_i)_i)$ is

$$R(f, (x_i)_i, (t_i)_i) = \sum_i f(t_i)(x_i - x_{i-1}).$$

A function f is *Riemann integrable*, of integral S , if

$$\begin{aligned} \forall \varepsilon > 0 \quad \exists \delta > 0 \quad \forall (x_i)_i, (t_i)_i \\ (\forall i \mid |x_i - x_{i-1}| < \delta) \implies |R(f, (x_i)_i, (t_i)_i) - S| < \varepsilon. \end{aligned}$$

The **generalized Riemann integral** replaces δ with a function $\delta : [a, b] \rightarrow \mathbf{R}^{>0}$, evaluated in t_i . The following functions are integrable:

- Riemann integrable functions: $\mathcal{R} \subset \mathcal{R}^*$;
- $\mathbf{1_Q}$;
- Lebesgue-integrable functions:

$$\mathcal{L} = \{f \in \mathcal{R}^* : |f| \in \mathcal{R}^*\};$$

- The derivative of (continuous) functions differentiable except at a countable number of points.

There is no need to add improper integrals: such functions are already integrable.

The convergence theorems from Lebesgue integration (uniform, monotone, dominated convergence) are still valid.

Regulated functions:

Bourbaki's alternative to the Riemann integral
S.K. Berberian (1979)

Another alternative (not a generalization) of the Riemann integral. *Regular functions* are functions with one-sided limits at each point; equivalently, they are uniform limits of step functions. For every regulated function f , there is a function F , differentiable except perhaps at countably many points, with $F' = f$ (the converse is not true). Regulated functions are Riemann integrable.

A friendly introduction to RGP

O. Flasch (2014)

The **rgp** package provides genetic programming (symbolic regression) and typed genetic programming (idem, with higher-order functions); it can be used with **emoa** for multiobjective optimization and **SPOT** for hyperparameter tuning. Alternatives include **ECJ** (Java), **GPTIPS** (Matlab) and, on the commercial side, **Data-Modeler**, **Discipulus** and **Eureqa**.

Gradient boosting machines, a tutorial

A. Natekin and A. Knoll (2013)

Boosting builds an ensemble model by fitting simplistic models (base learners, weak learners) to the data, with more weight on the observations currently misclassified by the ensemble. **Gradient boosting** gives an interpretation of this algorithm as a gradient descent, not

in parameter space, but in function space, *i.e.*, we do not look at

$$\sum_i \frac{\partial \text{Loss}(y_i, h(x_i, \theta))}{\partial \theta} \quad \text{but} \quad \sum_i \frac{\partial \text{Loss}(y, \hat{y})}{\partial \hat{y}} \bigg|_{\substack{y=y_i \\ \hat{y}=f(x_i)}}.$$

More precisely, we iterate:

$$\begin{aligned} (\rho_t, \theta_t) &= \underset{\rho, \theta}{\text{Argmin}} \sum_i \left[-\frac{\partial \text{Loss}(y, \hat{y})}{\partial \hat{y}} \bigg|_{\substack{y=y_i \\ \hat{y}=f_{t-1}(x_i)}} - \rho h(x_i, \theta) \right]^2 \\ f_t &\leftarrow f_{t-1} + \rho_t h(\cdot, \theta_t) \end{aligned}$$

The weak learners can be: linear regression, ridge regression, mixed model, splines, radial basis functions, GAM, tree stumps, trees with a maximum interaction depth (5 is good enough), wavelets (Viola-Jones face detection algorithm), etc.

The loss function can be: L^2 , L^1 , Huber, quantile loss, binomial loss $\log(1 - e^{-y\hat{y}})$, Adaboost loss $e^{-y\hat{y}}$, etc.

Generalizations include:

- If there are many variables, fit the base learners on a random subset of the variables, several times, and keep the best fit – this gives a sparser model;
- Subsampling (bagging);
- Shrinkage: $f_t \leftarrow f_{t-1} + \lambda \rho_t h(\cdot, \theta_t)$;
- Early stopping – this is needed; the optimal number of iterations depends on the step size λ and can be estimated by cross-validation.

Laplacian eigenmaps for dimensionality reduction and data representation

M. Belkin and P. Niyogi (2002)

Given a cloud of points x_1, \dots, x_N , build a graph, with those points as vertices, and an edge when the distance between two points is below some threshold, or when one is among the k nearest neighbours of the other; use 1 or $w_{ij} = \exp - \|x_i - x_j\|^2 / t$ as weights (similarities).

Spectral dimension reduction solves the generalized eigenvalue problem $Lf = \lambda Df$, where $D = \text{diag}(W\mathbf{1})$ and $L = D - W$,

$$Lf_i = \lambda_i Df_i \quad 0 = \lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_m,$$

and uses f_1, \dots, f_m as coordinates.

In dimension 1 (Fiedler vector), this can be justified as follows: we want a 1-dimensional embedding $(y_1, \dots, y_N) \in \mathbf{R}^N$ so that points that should be close (W_{ij} large) be close, e.g., by minimizing $\sum_{ij} (y_i - y_j)^2 W_{ij}$, with some normalization constraint, e.g., $\|y\|^2 = 1$ or $y'Dy = 1$ (in order to avoid the trivial solution $y = \mathbf{1}$, we also add $y \perp \mathbf{1}$, *i.e.*, $y'D\mathbf{1} = 0$ – this will give the smallest eigenvalue, skipping zero). Noticing that $\sum (y_i - y_j)^2 W_{ij} = y'Ly$, the optimization problem becomes

$$\begin{aligned} \text{Find} \quad & y \in \mathbf{R}^N \\ \text{To minimize} \quad & y'Ly \\ \text{Such that} \quad & y'Dy = 1 \text{ and } y'D\mathbf{1} = 0 \end{aligned}$$

and can be solved with Lagrange multipliers

$$\begin{aligned} f(y) &= y'Ly \\ g(y) &= y'Dy \\ \nabla f &= \lambda \nabla g, \end{aligned}$$

i.e., $Ly = \lambda Dy$ – a generalized eigenvalue problem.

This interpretation remains valid in a continuous setting, *i.e.*, if we replace the cloud of points with a compact manifold M :

$$\begin{aligned} \text{Find} \quad & f : M \rightarrow \mathbf{R}, \mathcal{C}^2 \\ \text{To minimize} \quad & \int \|\nabla f\|^2 \\ \text{Such that} \quad & \|f\|^2 := \int_M f^2 = 1 \\ & \langle f, 1 \rangle := \int_M f = 0 \end{aligned}$$

where $\|\nabla f\|$ estimates how far apart f maps nearby points. The objective can be written $\int \|\nabla f\|^2 = \int \langle \nabla f, \nabla f \rangle = \int \langle \nabla^* \nabla f, f \rangle = \int \mathcal{L} f \cdot f$, where $\mathcal{L} = \nabla^* \nabla = -\operatorname{div} \nabla$ is the Laplace-Beltrami operator (that is the name of the Laplace operator on a Riemannian manifold); it is positive semidefinite and (if M is compact) its spectrum is discrete $0 = \lambda_0 \leq \lambda_1 \leq \dots$. The first eigenvalues give a low-dimensional embedding.

To find two clusters in a weighted graph, one can try to find a partition $V = A \amalg B$ that minimizes

$$\operatorname{Cut}(A, B) = \sum_{\substack{u \in A \\ v \in B}} W_{uv},$$

but this tends to cut off weakly-connected outliers. Instead, one can minimize the normalized cut

$$\operatorname{NCut}(A, B) = \operatorname{Cut}(A, B) \left(\frac{1}{\operatorname{vol} A} + \frac{1}{\operatorname{vol} B} \right),$$

where $\operatorname{vol} A = \sum_{\substack{u \in A \\ v \in V}} W_{uv}$. By setting

$$x_1 = \begin{cases} 1/\operatorname{vol} A & \text{if } i \in A \\ 1/\operatorname{vol} B & \text{if } i \in B, \end{cases}$$

one can see that

$$\begin{aligned} \operatorname{NCut}(A, B) &= \frac{x' L x}{x' D x} \\ x' D \mathbf{1} &= 0; \end{aligned}$$

the relaxed minimum normalized cut problem, aka **spectral clustering**, is a generalized eigenvalue problem.

The local linear embedding (LLE) algorithm builds an unweighted graph from a cloud of points; the barycentric coordinates W_{ij} minimize $\sum_i \|x_1 - \sum_j W_{ij} x_j\|^2$, and are normalized by $\forall i \sum_j W_{ij} = 1$; the embedding is given by the k lowest eigenvalues of $E = (I - W)'(I - W)$. One can see that $E \approx \frac{1}{2} L^2$ (under some conditions); the eigenvectors are the same as those of L .

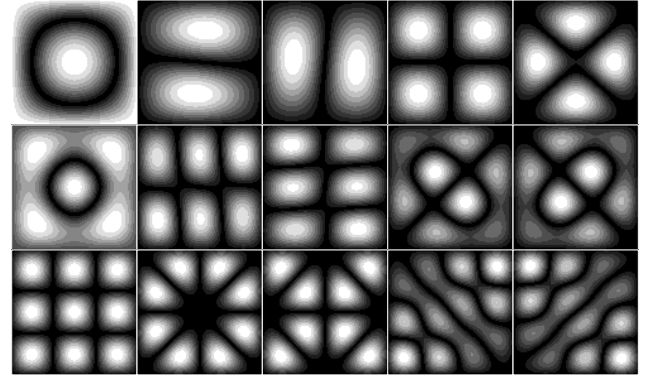
Constructing Laplace operator from point clouds in \mathbf{R}^d M. Belkin et al.

One can approximate the Laplace operator L on a submanifold $M \subset \mathbf{R}^d$ of (known) dimension k from a finite sample of points $P \subset M$ as follows:

- Approximate the tangent space at $p \in P$, for instance by the k plane Q^* through p best fitting $P_r = P \cap B(p, r)$, *i.e.*, minimizing the 1-sided Hausdorff distance $d_H(P_r, Q^*) = \sup_{p \in P_r} \inf_{q \in Q^*} d(p, q)$; an approximation \hat{T}_p of Q^* is sufficient;
- Build the Delaunay triangulation K_δ of the projection $\pi(P_\delta)$ or P_δ on \hat{T}_p ;
- For a function $f : P \rightarrow \mathbf{R}$, set $Lf(p)$ to

$$\sum_{\substack{\sigma: k\text{-dim} \\ \text{face of } K_\delta}} \frac{\operatorname{vol} \sigma}{k+1} \sum_{q \in V(\sigma)} [f(p) - f(\pi^{-1}q)] e^{-\frac{\|f(p) - f(\pi^{-1}q)\|^2}{4t}}.$$

Contrary to other implementations (e.g., the adjacency matrix with Gaussian weights), it does not assume that the points are drawn uniformly from M and does not require a global mesh (given a mesh, one can use the same formula, with the whole mesh instead of K_δ , and the tangent space is not needed ($\pi = \operatorname{id}$) – this is the **mesh-Laplace operator**).



Convergence, stability and discrete approximation of Laplace spectra T.K. Dey et al.

The mesh-Laplace operator gives a good approximation of the Laplace operator (pointwise convergence) and a good (accurate, robust) approximation of its spectrum.

The spectrum of the Laplacian gives some information on the manifold, e.g., its volume (Weyl's law: $\lambda_n \sim 4\pi n / \operatorname{Vol} M$) or its total curvature.

Differential representation for mesh processing O. Sorkine (2006)

The graph Laplacian of a mesh in \mathbf{R}^3 can be interpreted as the linear transformation from coordinates

to δ -coordinates (useful, e.g., for mesh editing)

$$\begin{aligned} L : v_i &\mapsto \delta_i = v_i - \frac{1}{|N(i)|} \sum_{j \in N(i)} v_j \\ &= \frac{1}{|N(i)|} \sum_{j \in N(i)} v_i - v_j, \end{aligned}$$

i.e., $L = I - D^{-1}A$, symmetrized $L_{\text{sym}} = DL = D - A$, $L_{\text{sym}}x = D\delta_x$. There are many variants, e.g.,

$$\begin{aligned} \delta_i &= \frac{1}{|\Omega_i|} \sum_j \frac{1}{2} (\cot \alpha_{ij} + \cot \beta_{ij}) (v_i - v_j) \\ \delta_i &= \frac{1}{|\Omega_i|} \sum_j \frac{\tan(\theta_{ij}^1/2) + \tan(\theta_{ij}^2/2)}{\|v_i - v_j\|} (v_i - v_j) \end{aligned}$$

where $|\Omega_i|$ is the volume of the Voronoi cell of i , α_{ij} , β_{ij} are the angles opposite the edge (ij) , θ_{ij}^1 , θ_{ij}^2 are the angles at i along (ij) .

Reconstructing the coordinates from the δ -coordinates (with a few positional “constraints” on some of the vertices) can be approached as a least squares problem.

If the mesh is known (e.g., a character in a video game – the mesh is just deformed as it moves), we can use the eigenfunctions of the Laplacian to encode the coordinates (it is not really necessary to compute the eigenfunctions: the reconstruction can also be reduced to a least squares problem).

Distilling the knowledge in a neural network **G. Hinton et al.**

To transfer the information in a complex model to a smaller model (model compression, model distillation), one often trains the smaller model to reproduce the output of the complex one on a very large (unlabeled) dataset. In the case of a multiclass classification problem, since there is information in the low probabilities, one can, instead of reproducing the output classes of the complex model, reproduce its probabilities or, rather, their logits. Even better, one can increase the temperature in the learning phase (to help learn smaller probabilities), and set it back to 1 afterwards.

A SQP method for general nonlinear programs using only equality constrained subproblems **P. Spellucci**

The `donlp2` algorithm solves a nonlinear program with equality and inequality constraints by taking a second order approximation of the objective, linearizing the constraints, replacing some inequalities with equalities and discarding the others. The subtleties reside in the choice of the inequalities to keep as equalities (those violated or near-active) or to discard, and how to deal with inconsistent quadratic programs.

Q-learning and SARSA: a comparison between two intelligent stochastic control approaches to financial trading **M. Corazza and A. Sangalli**

Misguided attempt to apply reinforcement learning (RL) to trading, using the past returns as state: they use the *past* Sharpe ratios as rewards – by construction, this will give a momentum strategy.

Universal value function approximators **T. Schaul et al. (2015)**

The value function approximator $V(s; \theta)$ learnt in reinforcement learning can be generalized to account for various goals (e.g., a single desired final state), $V(s, g; \theta)$, and used for planning (do not try to get to the final goal from the start; try to reach intermediate milestones first).

Weighted Voronoi stippling **A. Secord (2002)**

Stippling (non-photorealistic rendering) with weighted *centroidal Voronoi diagrams*, i.e., Voronoi diagrams in which the generators are the centroids of their cell.

Modern C **J. Gustedt (2015)**

Long presentation of C11.

Abstract tensor systems and diagrammatic representations **J. Lazovskis (2012)**

Formalization of Penrose’s diagrammatic tensor calculus, with a focus on Lie algebras.

Sampling for inference in probabilistic models with fast Bayesian quadrature **T. Gunter et al.**

Bayesian quadrature is a faster variant of MCMC. For positive integrands (e.g., a likelihood), use a Gaussian process prior on its square root or its logarithm.

Probabilistic line searches for stochastic optimization **M. Mahsereci and P. Hennig**

Stochastic gradient descent (SGD) has two problems:

- The gradient (even if it were not noisy) is not the best direction: use momentum, AdaGrad, etc.;
- One must choose the learning rate or resort to an expensive line search: instead of a line search, try Bayesian optimization.

Sequential Bayesian prediction in the presence of changepoints and faults **R. Garnett (2009)**

List of covariance structures to detect change points with Gaussian processes (GP):

- Drastic changepoint: the observations before and after are independent – the result is discontinuous;
- Continuous drastic changepoint: the observations before and after are independent conditionally on the value at the changepoint;
- Change in input and/or output scales,

$$K(x, y) = \lambda_{\text{out}}^2 \kappa \left(\frac{|y - x|}{\sigma_{\text{in}}} \right);$$

- Faults: bias, stuck value, drift, etc.

RRegrs: an R package for computer-aided model selection with multiple regression models
G. Tsiliki et al. (2015)

To estimate, compare and choose between 10 regression models: linear, lasso, elastic net, SVM, neural net, random forest, recursive feature elimination (fit the full model, rank the features, take the top k , re-fit the model, keep the best of those smaller models), etc., via `caret`.

Bellman's GAP: a 2nd generation language and system for algebraic dynamic programming
G. Sauthoff (2011)

Textbook dynamic programming (DP) is straightforward, but real-life DP can involve more tables, more recurrence relations, more indices (for instance, sequence alignment with affine gap penalty requires several tables). Algebraic dynamic programming (ADP) simplifies DP by describing the search space (e.g., the set of all possible alignments) with a grammar – this removes all the indices. “Bellman’s GAP” is a Java-like programming language for ADP. Alternatives include Haskell- or Ocaml-based DSLs and/or code generators and may not scale well.

Categorification of persistent homology
P. Bubenik and J.A. Scott (2014)

Persistent homology can be expressed in categorical terms. An ε -interleaving between $F, G : (\mathbf{R}, \leq) \rightarrow \mathcal{D}$ is a pair of natural transformations

$$\begin{array}{ccc} (\mathbf{R}, \leq) & & (\mathbf{R}, \leq) \\ \downarrow T_\varepsilon & \searrow F & \searrow G \\ (\mathbf{R}, \leq) & \xrightarrow{G} & \mathcal{D} \\ \downarrow T_\varepsilon & \searrow F & \searrow G \\ (\mathbf{R}, \leq) & \xrightarrow{G} & \mathcal{D} \end{array}$$

where T_ε is the translation, such that

$$\begin{array}{ccccc} F(a) & & & & F(b) \\ & \searrow & & \swarrow & \\ & G(a + \varepsilon) & \longrightarrow & G(b + \varepsilon) & \\ & \swarrow & & \searrow & \\ F(a + 2\varepsilon) & & & & F(b + 2\varepsilon) \\ & \searrow & & \swarrow & \\ & G(a + 3\varepsilon) & \longrightarrow & G(b + 3\varepsilon) & \end{array}$$

This defines an *interleaving distance*

$$d(F, G) = \inf \{ \varepsilon : F, G \text{ } \varepsilon\text{-interleaved} \}.$$

For any functor H , $d(HF, HG) \leq d(F, G)$. In $\mathcal{V}\text{ec}^{(\mathbf{R}, \leq)}$, *tame* diagrams (constant on a neighbourhood of each $t \in \mathbf{R}$, except a finite number of them) are of *finite type*; they can be described as *barcodes*. The *bottleneck distance* on barcodes defines an isometric embedding $\{\text{barcodes}\} \hookrightarrow \mathcal{V}\text{ec}^{(\mathbf{R}, \leq)}$. Given $f, g : X \rightarrow \mathbf{R}$ (not necessarily continuous), consider their lower level sets $F, G : (\mathbf{R}, \leq) \rightarrow \mathcal{T}\text{op}$; let $H : \mathcal{T}\text{op} \rightarrow \mathcal{D}$ be a functor; then, $d(HF, HG) \leq \|f - g\|_\infty$, i.e., H is stable to perturbations; this applies to persistent homology or relative homology. If \mathcal{D} is abelian, the category of ε -interleavings of diagrams $(\mathbf{R}, \leq) \rightarrow \mathcal{D}$ is also abelian.

Econometrics: methods and applications
P.H. Frances et al. (Coursera, 2015)

Linear models assume that the predictive variables are not stochastic. In the model $Y = X_1\beta_1 + X_2\beta_2 + \varepsilon$, the predictive variable X_1 is **endogenous** if $\text{Cor}(X_1, \varepsilon) \neq 0$: the ordinary least squares estimate $\hat{\beta}_1$ is then inconsistent. This can happen if a variable correlated with X_1 has been omitted, if there is a feedback mechanism between the predictive variables (e.g., when predicting sales from price and demand, it may seem that higher prices are good for sales, but higher prices may just come from higher demand), or if there are measurement errors (we do not observe X_1 but $X_1 + \eta$).

To address endogeneity, **2-stage least squares** (2SLS) looks for **instrumental variables** (IV) Z , i.e., variables Z that affect y only through X : $\text{Cor}(Z, X) \neq 0$, $\text{Cor}(Z, \varepsilon) = 0$ and uses them to find β : $\text{Cov}(Z, y) = \text{Cov}(Z, X)\beta$. The computations can also be done using two linear regressions ($X \sim Z$, then $y \sim \hat{X}$).

The method is consistent if $\frac{1}{n}Z'\varepsilon \rightarrow 0$ (Z and ε are uncorrelated), $\lim \frac{1}{n}Z'Z$ invertible (Z is not multicollinear), $\lim \frac{1}{n}Z'X$ has rank k (the number of variables – Z and X are sufficiently correlated). In particular, you need at least as many instruments as predictive variables (use the intercept and the exogenous variables as instruments: there will be fewer to find).

The **Sargan test** checks if $H_0 : \text{Cor}(Z, \varepsilon) = 0$ by approximating the residuals using 2SLS: estimate $\hat{X} = Z(Z'Z)^{-1}Z'X$ (1S), estimate $b = (\hat{X}'\hat{X})^{-1}\hat{X}'y$ (2S), compute the residuals $e = y - Xb$ (this is X , not \hat{X}), compute the R^2 of $e \sim Z$: under the null hypothesis, it is small, and $nR^2 \sim \chi^2(m - k)$ (where m is the number of instruments and k the number of variables in X).

The **Hausmann test** for “ $H_0 : X_1$ is actually exogenous” uses the R^2 of $\text{res}(y \sim X) \sim X + \text{res}(X_1 \sim Z)$: it should be small and $nR^2 \sim \chi^2(k_1)$.

The course also covered variable selection (forward or backward, using T or F tests, AIC, BIC, L^2 (RMSE) or L^1 (MAE) norm of the residuals), pseudo- R^2 for logistic regression (McFadden, Nagelkerke), time series (ARMA, unit roots, cointegration, Granger causality) and many regression tests:

- Regression specification (RESET): add the powers of

the fitted values $y_1 = x'_i\beta + \sum_{j=1}^p \gamma_j \hat{y}_i^{j+1} + \varepsilon_i$ (instead of the powers of x) and perform an F test for $H_0 : \forall j \gamma_j = 0$ (it is approximate because the \hat{y}_i are not fixed predictors);

- Chow break test: split the data in two, at a potential break point, and use an F test for $H_0 : \beta_{\text{before}} = \beta_{\text{after}}$;
- Chow forecast test: linear before the break, perfect fit after: $y_i = x'_i\beta + \sum_{j=n_1+1}^{n_1+n_2} \gamma_j \delta_{ji} + \varepsilon_i$, $H_0 : \forall j \gamma_j = 0$;
- Jarque-Bera normality test (for the residuals): some combination of the sample skewness and kurtosis of a Gaussian variable is approximately $\chi(2)$ -distributed.

Integer relation detection **D.H. Bailey (2000)**

Given x_1, \dots, x_n (high-precision) floating-point numbers, the **PSLQ** algorithm looks for integers k_1, \dots, k_n , not all zero, so that $\sum_i k_i x_i = 0$. Given a number α of interest, apply it to:

- $1, \alpha, \alpha^2, \dots, \alpha^n$ to show that α is a root of a simple polynomial;
- $\alpha, \zeta(1), \zeta(2), \dots, \log 2, \log 3, \dots, \pi, \pi^2, \dots, \text{Li}_k(\ell), \pi^k \zeta(\ell)$, etc. to find interesting relations.

It was also used to find a base-16 expansion of π .

For n numbers and integer coefficients with at most d digits, you need at least nd significant digits.

The LLL algorithm has similar applications.

It is available in a C++ library, in GAP, perhaps also from SAGE via `mpmath` and/or `GMP`.

The decompositional approach to matrix computation **G.W. Stewart (2000)**

The “big six” matrix decompositions are:

- Cholesky: $A = RR'$ (or $A = LDL'$), where A is positive definite and R upper-triangular; used to solve $A'x = b$ or compute $x'A^{-1}x$;
- LU: $P'AQ = LU$, where A is square, L lower-triangular, U upper-triangular, P and Q permutations; used to solve $Ax = b$;
- QR: $A = QR$, where Q is orthogonal and R upper-triangular with non-negative diagonal elements; used in least-squares estimation: QQ' is the projection on $\text{Im } A$;
- Spectral decomposition: $A = V\Lambda V'$, where A is symmetric, V orthogonal and Λ diagonal;
- Schur: $A = UTU^*$, where A is square, U unitary, T upper-triangular; sometimes an intermediate step in eigenvalue problems;
- SVD: $A = U\Sigma V'$, where U and V have orthogonal columns and Σ is diagonal.

Using multi-complex variables for automatic computation of high-order derivatives **G. Lantoin et al (2012)**

The first derivative of an analytic function $f : \mathbf{R} \rightarrow \mathbf{R}$ can be approximated as $f'(x) = \text{Im} \frac{f(x+ih)}{h} + o(h)$. For higher derivatives, use **multicomplex numbers**,

$$\mathbf{C}_n = \frac{\mathbf{R}[i_1, \dots, i_n]}{(i_1^2 + 1, \dots, i_n^2 + 1)}$$

$$f^{(n)}(x_0) \approx \text{Im}_{1,2,\dots,n} \frac{f(x_0 + hi_1 + \dots + hi_n)}{h^n}$$

For higher derivatives, some have also suggested to use Cauchy’s formula,

$$\oint_C \frac{f(z)}{z - z_0} dz = 2\pi i f(z_0),$$

which generalized to

$$f^{(n)}(z_0) = \frac{n!}{2\pi i} \oint_C \frac{f(z)}{(z - z_0)^n} dz.$$

Learning to discover efficient mathematical identities **W. Zaremba et al.**

The set of equivalent reformulations of a mathematical expression (e.g., `sum((A*B)^6)`, where A and B are matrices) form a graph, with edges corresponding to valid transformations. Computer algebra systems (Maple, Mathematica) use a set of heuristic rules to explore this graph and return a simpler expression (either a shorter one, or one with a lower algorithmic complexity). One can also use random exploration.

Instead, machine learning (e.g., n -grams, or RNN) can guide the exploration.

A contextual-bandit approach to personalized news article recommendation **L. Li et al.**

There are 3 classes of bandit algorithms:

- ε -greedy (with decaying ε);
- Upper confidence bounds (UCB);
- Bayesian approaches (Gittins index).

LinUCB generalizes UCB to contextual bandits: if the reward is linear, i.e., $E[r_a|x_a] = x'_a \theta_a$, θ_a unknown, a = arm, a confidence interval can be computed in closed form.

Frequentism and Bayesianism: a Python-driven primer **J. VanderPlas (2014)**

Frequentist confidence intervals can be problematic: for instance, for the truncated exponential $P(x|\theta) = e^{\theta-x} \mathbf{1}_{x>\theta}$ and the observed data $\{10, 12, 15\}$, the 95% confidence interval is $(10.2, 12.2)$, even though we know that $\theta < 10$ – the corresponding bayesian **credible interval** is $(9.0, 10.0)$.

The paper also details Bayesian linear regression, with `emcee` (affine invariant ensemble MCMC), `PyMC` (Metropolis-Hastings) and `PyStan` (Hamiltonian

MCMC), and explains how to choose an uninformative prior: we want it to be invariant under the transformations $(x, y) \mapsto (y, x)$ and $(x, y) \mapsto (\lambda x, \lambda y)$ – this second transformation gives the **Jeffreys prior** for the variance, $P(\sigma) \propto 1/\sigma$.

**An object-oriented framework
for robust multivariate analysis**
V. Todorov and P. Filzmoser (2009)

The **rrcov** package provides robust estimators of scale and location:

- **Minimum covariance determinant (MCD)**: mean and variance of the k observations whose covariance matrix has the smallest determinant (heuristically: start with k points, compute $\hat{\mu}$ and \hat{V} , $d_i^2 = (x_i - \hat{\mu})' \hat{V}^{-1} (x_i - \hat{\mu})$, take the k points with the smallest distance, iterate);
- **Minimum volume ellipsoid (MVE)** that contains at least half the data;
- **Stahel-Donoho**: weighted mean and covariance,

$$w_i = \text{Min} \left\{ 1, \left(\frac{c}{r_i} \right)^2 \right\}$$

$$r_i = \text{Max}_a r(x_i, a) \quad \text{outlyingness}$$

$$r_i(x_i, a) = \frac{|x_i' a - m(a' X)|}{s(a' X)} \quad \text{outlyingness in direction } a$$

m, s : robust 1-dimensional location and scale;

- **Orthogonalized Gnanadesikan-Kettenring (OGK)**: given a robust 1-dimensional scale σ , define robust covariances

$$s_{ij} = \frac{1}{4} \left[\sigma \left(\frac{X_i}{\sigma X_i} + \frac{X_j}{\sigma X_j} \right)^2 - \sigma \left(\frac{X_i}{\sigma X_i} - \frac{X_j}{\sigma X_j} \right)^2 \right]$$

and tweak the resulting covariance matrix to make it positive semidefinite;

- **S-estimator**: solution of the optimization problem

$$\begin{array}{ll} \text{Find} & \mu, V \\ \text{To minimize} & \sigma(d_1, \dots, d_n) \\ \text{Such that} & \det V = 1 \end{array}$$

where $d_i = (x_i - \mu)' V^{-1} (x_i - \mu)$, and σ is an M-estimator of scale, *i.e.*, $\frac{1}{n} \sum \rho(z/\sigma) = \delta$, $\delta \in (0, 1)$, $\rho : \mathbf{R}_+ \rightarrow [0, 1]$ increasing from 0 to 1.

It also provides **robust PCA**:

- PCA using a robust variance matrix;
- Projection pursuit, *i.e.*, finding the direction in which some quantity (a robust 1-dimensional estimator of scale) is maximal;
- Huber's method: consider all the directions defined by pairs of points, and project the data on those directions; for each point and each of those directions, compute the normalized distance to the center, using 1-dimensional robust estimators of location and scale; use the k points with the lowest distance.

Robust principal component analysis?
E.J. Candès (2009)

If a matrix is the sum of a (dense) low-rank and a sparse matrix, $M = L + S$, the components can be recovered by solving the convex optimization problem

$$\begin{array}{ll} \text{Find} & L, S \\ \text{To minimize} & \|L\|_* + \lambda \|S\|_1 \\ \text{Such that} & L + S = M, \end{array}$$

where $\|\cdot\|_*$ is the nuclear norm (sum of the singular values). Depending on the application, L or S may be of interest.

**Introduction to numerical methods
in differential equations**
M.H. Holmes (2007)

1. To solve an *initial-value problem* (IVP), *i.e.*, an ordinary differential equation (ODE) of the form $y' = f(y, x)$, $y(0) = \alpha$, one can discretize the derivative. There are many ways of doing that (forward Euler, backward Euler, centered, leapfrog, etc.) and the resulting numeric schemes have different stability properties.

A numeric scheme is **A-stable** if the numeric solution of $y' = -ry$, $y(0) = \alpha$ remains bounded (the exact solution decays exponentially). This equation is chosen because, near a stable (constant) solution, all ODEs look like that. A scheme is *conditionally A-stable* if it is stable when the step size is sufficiently small. It is *strictly A-stable* if the solution converges to zero. It is *monotone A-stable* if the solution is monotonic. The forward (Euler) scheme is conditionally (monotone) A-stable and explicit; the backward scheme is (monotone) A-stable but implicit; the leapfrog method is unstable.

One can also integrate the equation, $y_{i+1} - y_i = \int_{t_i}^{t_{i+1}} f(y, t) dt$ and use numeric integration, *e.g.*, left box, right box, midpoint, trapezoidal rule (implicit, A-stable, $O(h^2)$, conditionally monotone), Simpson's rule, etc. The Adams method approximates f with a function that can be integrated exactly (for instance, a quadratic approximation gives Simpson's rule).

The trapezoidal method is $O(h^2)$, but it is implicit.

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{2} h (f_i + f_{i+1}) \\ f_i &= f(t_i, y_i) \\ f_{i+1} &= f(t_{i+1}, y_{i+1}). \end{aligned}$$

It can be turned into an explicit method by replacing f_{i+1} , which we do not know yet, by its Euler approximation: the second order *Runge-Kutta* (RK) method (Heun) is $O(h^2)$, explicit, conditionally A-stable.

$$\begin{aligned} y_{i+1} &= y_i + \frac{1}{2} (k_1 + k_2) \\ k_1 &= h f(t_i, y_i) \\ k_2 &= h f(t_{i+1}, y_i + k_1) \end{aligned}$$

The popular RK4 method uses a similar idea, with Simpson's rule.

Newton's equation, $my'' = F(y)$, conserves energy, $H(t) = \frac{1}{2}my'^2 + V(y)$, where $V'(y) = -F(y)$, but not all numeric schemes do. The trapezoidal method does, but it is implicit. The *velocity Verlet method* simply replaces it with its Euler approximation (it is preferred to RK4 in physics). It does not exactly conserve energy, but almost: it is *symplectic*.

2. A *boundary value problem* (BVP) is an ODE of the form $y'' = f(x)$, $y(0) = \alpha$, $y(1) = \beta$.

If the equation is linear, $y'' + p(x)y' + q(x)y = f(x)$, the centered approximations of y' and y'' give a tridiagonal system (choose a step size $h < 2/\|p\|_\infty$). If the condition number is 10^n (it is $O(N^2)$, where N is the number of steps), do not expect more than $15 - n$ significant digits.

The discretization of non-linear BVP gives a non-linear system of equations, which can be solved by Newton's method (the Jacobian is tridiagonal); the solution need not be unique.

Residual methods look for solutions of the form $y(x) = \sum a_k \phi_k(x)$, for some basis functions ϕ_k (B-splines, Fourier) so that the residual be zero on a grid (*collocation*) or minimal in L^2 norm (least squares).

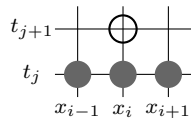
Shooting methods solve the IVP $y(0) = \alpha$, $y'(0) = s$, and adjust s so that $y(1) = \beta$, hoping the problem is not ill-conditioned.

3. The heat equation

$$\begin{aligned} D \frac{\partial^2 u}{\partial x^2} &= \frac{\partial u}{\partial t} \\ u(0, t) &= u_{\text{left}}(t) \\ u(1, t) &= u_{\text{right}}(t) \\ u(x, 0) &= g(x) \end{aligned}$$

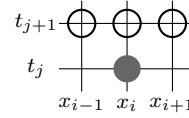
is a parabolic equation: it is a *diffusion* (it tends to smooth the initial condition – g need to be continuous, but u will), it satisfies the *maximum principle* (the maximum and minimum of u are on the boundary) and has the *instant messaging* property (even if $u(\cdot, 0)$ is zero on some interval, it immediately becomes non-zero).

The explicit method (forward difference for time, centered for space), with *stencil*

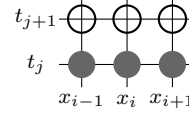


does not satisfy the instant messaging property and has stability problems as t increases. (PDEs have ad hoc notions of stability, e.g., by looking at how an oscillatory initial condition evolves; the explicit method requires $k = O(h^2)$, where k is the time step and h the space step.) The implicit method (backward dif-

ference)



has the instant messaging property and is stable. Numerical simulations suggest that the exact solution often lies between the explicit and implicit ones. The *theta method* mixes a proportion θ of explicit and $1 - \theta$ of implicit.



For $\theta = \frac{1}{2}$ (Crank-Nicolson – the most widely used method for parabolic equations), the error is $O(h^2) + O(k^2)$ (instead of $O(h^2) + O(k)$). The trapezoidal rule (for time) also gives the Crank-Nicolson method. It is not robust to the presence of jumps.

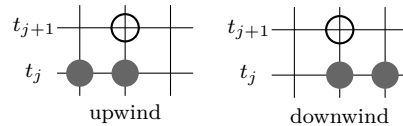
The *method of lines* discretizes only the space: this gives a family of coupled ODEs.

Residual methods, approximating the solution as $u(x, t) = \sum q_k(t)B_k(x)$, can also be used.

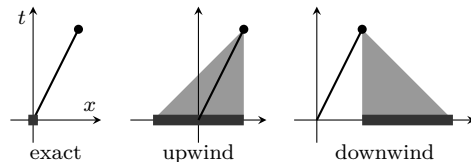
4. The advection equation is

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0, \quad u(x, 0) = g(x), \quad a > 0.$$

It can be solved exactly with a change of variables: $u(x, y) = g(x - at)$. Contrary to the heat equation, jumps in the initial condition g are preserved (strictly speaking, this is a weak solution). The upwind and downwind explicit schemes (forward difference in time, backward or forward in space)



have different *domains of dependence*.

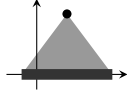


The downwind scheme will not work. For the upwind one, the exact domain of dependence is included in the numeric one (Courant-Friedrichs-Lewy (CFL) condition – this did not happen with the heat equation) if $ak/h \leq 1$. More generally, one could look at methods of the form $u_{i,j+1} = Au_{i+1,j} + Bu_{i,j} + Cu_{i-1,j}$, compute a Taylor expansion of the error and set to zero as many terms as possible. Implicit methods tend to have an infinite domain of dependence, which is not desirable for advection equations.

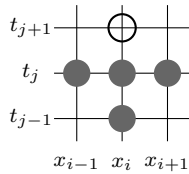
5. The wave equation is

$$\begin{aligned} c^2 \frac{\partial^2 u}{\partial x^2} &= \frac{\partial^2 u}{\partial t^2} \\ u(0, t) &= u(1, t) = 0 \\ u(x, 0) &= f(x) \\ u_t(x, 0) &= g(x). \end{aligned}$$

It can be solved by separation of variables (Fourier series) or by a change of variable. Jumps in the initial conditions are preseved. The domain of dependence is



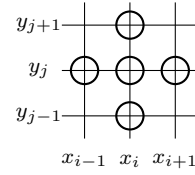
To study the stability of hyperbolic equations (or of their discretizations), look for solutions of the form $u(x, t) = e^{i(kx - \omega t)}$. For the wave equation, this gives $\omega = \pm ct$. For other equations (Klein-Gordon, $c^2 u_{xx} = u_{tt}$; modified advection, $u_t + au_x + bu = 0$; beam, $u_{xxxx} + u_{tt} = 0$; advection-diffusion, $Du_{xx} = u_t + au_x$; etc.), it may be complex, $u(x, t) = e^{at} e^{i(kx - bt)}$. The equation is *stable* if $\text{Im } \omega \leq 0$ (for all k), *non-dispersive* if $\text{Re } \omega \propto k$ (the speed of the wave does not depend on the wavelength), *non-dissipative* if $\text{Im } \omega = 0$ (for all k). The explicit method (centered differences)



satisfies the CFL condition if $\lambda = ck/h \leq 1$; it is non dissipative for $\lambda \leq 1$ but dispersive for $\lambda < 1$ (problematic for short wave lengths).

6. The Laplace equation $\Delta u = 0$, the prototypal elliptic PDE, is the steady state of the wave equation: it has similar smoothing properties. It can be solved, with a finite difference approximation (centered differ-

ences: 5-point scheme).



The resulting matrix has the following sparsity pattern, where T is tridiagonal and D diagonal.

$$\begin{pmatrix} T & D & 0 \\ D & & \\ 0 & D & T \end{pmatrix}$$

One can solve $Ax = b$, without inverting A , if A is symmetric positive definite, by gradient descent, minimizing $F(x) = \frac{1}{2}x'Ax - bx$:

$$\begin{aligned} x_{k+1} &\leftarrow x_k + \alpha_k d_k \\ d_k &: \text{descent direction} \\ r_k &= b - Ax_k = r_{k-1} - \alpha_{k-1} q_{k-1} \\ q_k &= Ad_k \\ \alpha_k &= \underset{\alpha}{\text{Argmin}} F(x_k + \alpha d_k) = \frac{d_k \cdot r_k}{d_k \cdot q_k}. \end{aligned}$$

Steepest descent, $d_k = -\nabla F(x_k) = r_k$, may converge very slowly if the level surfaces of F are eccentric ellipsoids, *i.e.*, $\text{cond } A \gg 1$ (but $\text{cond } A \approx 4N^2/\pi^2 \gg 1$). *Conjugate gradient* addresses this problem by using $d_k = r_k + \beta_{k-1} d_{k-1}$, where β is such that $\forall i \neq j$, $r_i \perp r_j$, *i.e.*, $\beta = \|r_{k+1}\|^2 / \|r_k\|^2$; the solution is reached in (at most) N steps.

To address the conditioning problem, find B invertible such that $\text{cond } BA \approx 1$ and solve $BAx = Bb$. To apply the conjugate gradient method, we need a symmetric (positive definite) matrix: just rewrite the equation as $(BAB')B'^{-1}x = Bb$. Popular choices for this *preconditioner* include $M = B^{-1}B'^{-1} = \text{diag } A$, $\text{tridiag } A$ and $(D+L)D^{-1}(D+U)$ where $A = D+L+U$, D diagonal, L and U strictly upper and lower triangular.

**Probabilistic numerics
and uncertainty in computations**
P. Hennig et al. (2015)

Many numeric (non-probabilistic) algorithms can be given a Bayesian flavour: Bayesian optimization is the best known, but not the only one.

In **Bayesian quadrature**, to estimate an integral, e.g., $\int_0^1 f$, one models the integrand as a Gaussian process $f \sim \text{GP}$: since integration is linear, $(\int_0^1 f, f(t_1), \dots, f(t_n))$ is Gaussian, and one can estimate the conditional distribution $\int_0^1 f \mid f(t_1), \dots, f(t_n)$. This gives an estimate of the integral and of its precision – but it depends a lot on (the smoothness of) the prior. One can also choose the point where to evaluate the function next to increase the precision the most. The prior (the covariance function of the Gaussian process) specifies the smoothness of the integrand (usually \mathcal{C}^∞) and perhaps other properties (e.g., the sign – but the distributions would no longer be Gaussian). Different priors lead to different quadrature rules.

$$f \sim \text{GP}(m, C) \implies \int f \sim N(\int m, \int \int C)$$

Bayesian quadrature can be generalized to **Bayesian ODEs** or PDEs.

Bayesian decision theory uses Bayesian quadrature to estimate and maximize expected utility. (Bayesian optimization uses Bayesian decision theory to choose the next point to evaluate.)

Bayesian linear algebra tries to solve $Ax = b$ where A is too large to be inverted, using the value of some products $As_1 = y_1, \dots, As_n = y_n$ to gain some knowledge about A^{-1} and $A^{-1}b$. The inverse A^{-1} is not known but can be modeled as a Gaussian variable; $(A^{-1}b, A^{-1}y_1, \dots, A^{-1}y_n)$ is then Gaussian, and so is the conditional distribution $A^{-1}b \mid s_1 = A^{-1}y_1, \dots, s_n = A^{-1}y_n$.

**Sampling for inference in probabilistic models
with fast Bayesian quadrature**

T. Gunter et al.

Bayesian quadrature, to estimate expectations $E[\ell] = \int \ell(x)\pi(x)dx$, with a Gaussian process (GP) prior on the integrand ℓ , is problematic when ℓ is a likelihood: the prior does not enforce non-negativity, and struggles to model the high dynamic range of most likelihoods. A Gaussian prior on the log-likelihood is better, but the integral is far from Gaussian. A square root transform, $\ell(x) = \alpha_f(x)^2$, $f \sim \text{GP}$, is a good compromise; the integral is not Gaussian but can be approximated by a Gaussian, either by linearization or moment matching (warped sequential active Bayesian integration, WS-ABI).

Projection pursuit
P.J. Huber (1985)

Given a random variable $X \sim f$ on \mathbf{R}^n , **projection pursuit** looks for a projection $X \mapsto AX$ on a low-dimensional subspace (dimension 1, 2 or 3) that maximizes (or minimizes) some *projection index* $Q(AX)$. This projection index may be:

- Equivariant (mean or some other measure of location);
- Location-invariant and scale-equivariant (standard deviation or some other measure of dispersion – this gives PCA and robust PCA);
- Affine-invariant, e.g., normality test statistics (uninteresting projections tend to be more Gaussian than those with idiosyncrasies): |skewness|, excess kurtosis (or other standardized absolute cumulants – but they tend to be very outlier-sensitive), standardized Fisher information

$$Q(X) = \sigma^2(X) \int \left(\frac{f'}{f} - \frac{\phi'}{\phi} \right)^2 f,$$

standardized negative Shannon entropy

$$Q(X) = - \int \log \left(\frac{\phi}{f} \right) f,$$

Kolmogorov-Smirnov, Durbin-Watson, etc. (this is now called independent component analysis, ICA).

The k -dimensional projection can be approximated in a greedy (stepwise) fashion (if you want a more precise result, use *backfitting*: once you have a k -dimensional projection, remove one vector, find its optimal replacement, and iterate until convergence).

Given two random variables X, Y in \mathbf{R}^n , a discriminating hyperplane can be found by using the T -statistic as a projection index, or

$$\frac{\text{ave}[a'X] - \text{ave}[a'Y]}{\text{sd}[a'(X \cup Y)]}$$

or

$$\frac{\text{med}[a'X] - \text{med}[a'Y]}{\text{mad}[a'(X \cup Y)]}.$$

The same idea gives a measure of outlyingness

$$r_i = \sum_a \frac{a'x_i - \text{med}[a'X]}{\text{mad}[a'X]}$$

which can be used to define weights and robust (weighted) estimators of location and scale.

Regression is the estimation of $f(x) = E[Y|X = x]$. **Projection pursuit regression** (PPR) estimates it as a sum of *ridge functions*

$$f(x) \approx \sum g_j(a'_j x)$$

(note that the ridge functions are constant on hyperplanes). The ridge functions can be estimated greedily:

- Compute the residuals $r_i = y_i - \sum_1^{m-1} g_j(a'_j x_i)$;
- Smooth the scatterplot $r_i \sim a'x_i$:

$$r_i = g(a'x_i) + \text{noise}$$

(this depends on a);

- Find the projection a that minimizes

$$\sum_i (r_i - g(a'x_i))^2;$$

- Iterate.

Not all functions can be represented exactly as a finite sum of ridge functions (but they can be approximated – and, even if it is possible, the decomposition need not be unique).

Projection pursuit density approximation (PPDA) uses the same idea to approximate probability distribution functions, with a multiplicative decomposition (to ensure non-negativity)

$$f(x) \approx f_0(x) \prod_1^k h_j(a'_j x),$$

for some reference distribution f_0 . One can either look for

$$f_k(x) = f_0(x) \prod_1^k h_j(a'_j x) \xrightarrow[k \rightarrow \infty]{} f(x)$$

or, dually,

$$f_{-k}(x) = f(x) \prod_1^k h_j(a'_j x) \xrightarrow[k \rightarrow \infty]{} f_0(x)$$

(this removes the features of f , one by one, until we are only left with the reference (featureless) f_0). The quality of the approximation can be measured with the *relative entropy* $E(f, g) = \int \log(f/g)$, the *Hellinger distance* $H(f, g) = \int (\sqrt{f} - \sqrt{g})^2$, the Prohorov distance

$$\pi(\mu, \nu) = \inf\{\varepsilon > 0 : \forall A \quad \mu(A) \leq \nu(A_\varepsilon) \text{ and } \nu(A) \leq \mu(A_\varepsilon)\},$$

the bounded Lipschitz metric

$$d(\mu, \nu) = \sup \left\{ \left| \int f d\mu - \int f d\nu \right| : f \in \mathcal{C}^0, \sup |f| \leq 1, \sup_{x \neq y} \left| \frac{f(x) - f(y)}{x - y} \right| \leq 1 \right\},$$

etc. One can proceed iteratively: find a minimizing $E(f_a, g_a)$ (where f_a and g_a are the marginal distributions and $g = f_0$); replace f with $f g_a / f_a$ (resp. g with $g f_a / g_a$); iterate.

Projection pursuit density estimation (PPDE) applies the same idea to density estimation (from samples):

- Standardize, using robust location and dispersion estimates;
- Choose f_0 with fat tails; fit it to the cloud of points;
- Proceed as above; if the marginal densities are too time-consuming to estimate, use sample-based estimates.

(Only use when there is not enough data or too many dimensions for a kernel estimator.)

Deconvolution tries to recover a signal x from a convolution $y = f * x$ with an unknown filter f . *Minimum entropy deconvolution* considers segments of length d , (y_t, \dots, y_{t+d-1}) , as points of \mathbf{R}^d , and finds the least Gaussian 1-dimensional projection $q' \cdot y_{t:t+d-1}$: q is an estimate of f^{-1} .

If a time series X_t is a sum of periodic signals with independent periods, they can be recovered by finding p maximizing

- $Q(p) = \text{ave}_t[Z_{p,t}^2]$, where $Z_{p,t} = \text{ave}_k[X_{t+kp}]$;
- Or $|C(p)|^2 = |\text{ave}_t[e^{2\pi i t/p} X_t]|^2$.

Reinforcement Learning D. Silver (2015)

1. Reinforcement learning (RL) is the search for the optimal policy for a Markov decision process (MDL) observed only through its history (past states, actions and rewards). A RL agent can model one or several of:

- The MDP itself;
- The value of each state;
- The policy.

Planning refers to the special case where the MDP is known.

2. A **Markov reward process** is a Markov process (a Markov chain) with rewards. The value of a state is

$$v(s) = E[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s],$$

i.e., $v = R + \gamma P v$.

A **Markov decision process** (MDP) adds actions. We still have $v = R + \gamma P v$, but v , R and P depend on the policy π . The value of a state, or of a state-action pair, for the optimal strategy (there exists an optimal deterministic strategy) satisfies the Bellman equation:

$$v_*(s) = \text{Max}_a q_*(s, a)$$

$$q_*(s, a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a v_*(s').$$

3. To find the optimal strategy for a known MDP (planning), **policy iteration** starts with an arbitrary policy, evaluates the values of the states for this policy, computes the greedy policy for those values, and iterates. Each step improves the policy and, in the limit, the Bellman equation is satisfied: it converges to the optimal policy.

Value iteration is a variant in which we do not compute the values exactly, but just iterate $v \leftarrow R + \gamma P v$ once. There is actually no need to keep track of the policy, it suffices to iterate on the values:

$$v(s) \leftarrow \text{Max}_a R_s^a + \gamma \sum_{s'} P_{ss'}^a v(s).$$

There are many variants:

- Update the states one at a time, in a random order;
- Only update the most promising states;
- Use a sample of the transition matrix;
- Focus on states close to the agent.

4. To estimate the value function of a given policy on an unknown MDP (*prediction*), **Monte Carlo RL** uses the average return (the sum of the discounted rewards) from complete episodes (*i.e.*, you have to let the MDP run until it reaches a final state to know the return). **Temporal difference learning** (TD) uses incomplete episodes:

$$\text{MC: } v(s_t) \leftarrow v(s_t) + \alpha(G_t - v(s_t))$$

$$\text{TD: } v(s_t) \leftarrow v(s_t) + \alpha(R_{t+1} + \gamma v(s_{t+1}) - v(s_t)).$$

MC is unbiased but has high variance; TD is biased but has lower variance. Instead of looking one step ahead, we can look n steps ahead and replace G_t with

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n v(s_{t+n}).$$

Forward-view TD(λ) combines them all:

$$G_t^\lambda = (1 - \lambda) \sum_{n \geq 1} \lambda^{n-1} G_t^{(n)} \quad (\lambda\text{-return})$$

$$v(s_t) \leftarrow v(s_t) + \alpha(G_t^\lambda - v(s_t)).$$

Backward-view TD(λ) only uses information from the past, by assigning credit for the reward to the most recent and frequent states, using **eligibility traces**.

$$E_0(s) = 0$$

$$E_t(s) = \gamma \lambda E_{t-1}(s) + \mathbf{1}_{s_t=s}$$

$$\delta_t = R_{t+1} + \gamma v(s_{t+1}) - v(s_t) \quad \text{1-step-ahead error}$$

$$v(s) \leftarrow v(s) + \alpha \delta_t E_t(s)$$

Backward- and forward-view TD(λ) are equivalent.

5. *Control* looks for the optimal value (or policy) of an unknown MDP. It can be *on-policy* (learn about the optimal policy while following it) or *off-policy* (learn about the optimal policy while following another one). Since the model is unknown, the *q-value function* is more useful than the value function to compute the policy.

$$\pi(s) = \underset{a}{\text{Argmax}} R_s^a + P_{ss'}^a v(s') \quad (P \text{ is unknown})$$

$$\pi(s) = \underset{a}{\text{Argmax}} q(s, a)$$

MC- ε -greedy control uses MC to evaluate the *q*-value function and ε -greedy exploration (with probability

$1 - \varepsilon$, choose the greedy (optimal) policy; with probability ε , act at random; update the strategy at the end of each episode).

SARSA control uses SARSA (TD for the *q*-value function) and ε -greedy exploration

$$q(s, a) \leftarrow q(s, a) + \alpha(R + \gamma q(s', a') - q(s, a)).$$

As before, these can be generalized to forward-view SARSA(λ)

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n v(s_{t+n})$$

$$q_t^\lambda = (1 - \lambda) \sum \lambda^{n-1} q_t^{(n)}$$

$$q(s_t, a_t) \leftarrow q(s_t, a_t) + \alpha(q_t^\lambda - q(s_t, a_t))$$

and backward-view SARSA(λ)

$$E_0(s, a) = 0$$

$$E_t(s, a) = \gamma \lambda E_{t-1}(s, a) + \mathbf{1}_{s_t=s, a_t=a}$$

$$\delta_t = R_{t+1} + \gamma q(s_{t+1}, a_{t+1}) - q(s_t, a_t)$$

$$q(s, a) \leftarrow q(s, a) + \alpha \delta_t E_t(s, a)$$

The step sizes usually decrease, but not too quickly (Robbins-Monro sequence: $\sum \alpha_t = \infty$, $\sum \alpha_t^2 < \infty$).

Off-policy algorithms learn about a policy π while following another policy μ . **Importance sampling MC**

$$G_t^{\pi/\mu} = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} \frac{\pi(a_{t+1}|s_{t+1})}{\mu(a_{t+1}|s_{t+1})} \dots \frac{\pi(a_T|s_T)}{\mu(a_T|s_T)} G_t$$

$$v(s_t) \leftarrow v(s_t) + \alpha(G_t^{\pi/\mu} - v(s_t))$$

has a high variance.

For **importance sampling TD**,

$$v(s_t) \leftarrow v(s_t) + \alpha \left(\frac{\pi(a_t|s_t)}{\mu(a_t|s_t)} (R_{t+1} + \gamma v(s_{t+1})) - v(s_t) \right),$$

the variance remains reasonable.

Q-learning does not use importance sampling, but uses π (not μ) in the target

$$q(a_t, s_t) \leftarrow q(a_t, s_t) + \alpha(R_{t+1} + \gamma q(s_{t+1}, a') - q(s_t, a_t))$$

where a' comes from π and a_{t+1} from μ . In particular, when $\pi = \text{greedy}$ and $\mu = \varepsilon\text{-greedy}$, this becomes **SARSAMAX** (or “Q-learning”):

$$q(s, a) \leftarrow q(s, a) + \alpha(R + \gamma \text{Max}_{a'} q(s', a') - q(s, a)).$$

Algorithm	Model	Policy	Aim	on-/off-policy
Policy iteration Value iteration	known	unknown	π^*	
MC TD(λ)	unknown	known	v or q	on
MC + ε -greedy TD(λ) + ε -greedy	unknown	unknown	q^*	on
IS MC or TD Q-learning				off

6. If the number of states is large, we cannot use a table lookup to model the value function $v(s)$ or $q(s, a)$: one can summarize the state as a set of features and use them in a linear (or nonlinear) model. This model attempts to minimize

$$J(w) = E_{\pi}(v_{\pi}(s) - v(s, w))^2,$$

e.g., with stochastic gradient descent (SGD),

$$\Delta w = \alpha(v_{\pi}(s) - v(s, w))\nabla J(w),$$

where, for a linear model $v(s, w) = x(s)'w$ with features $x(s)$, the gradient is $\nabla J(w) = x(s)$, i.e.,

$$\Delta w = \text{step size} \times \text{prediction error} \times \text{feature value}.$$

Since $v_{\pi}(s)$ is unknown, we use a target instead: the return G_t for MC, $R_{t+1} + \gamma v(s_{t+1}, w)$ for TD(0), the forward or backward λ -return G_t^{λ} for TD(λ).

Deep Q-networks (e.g., those that play Atari games) keep all the history and replay it (random mini-batches); they compute the Q-learning targets with some old, fixed (slowly updated) estimate of the q -function.

7. Instead of the value function, one can directly learn a parametrized policy $\pi_{\theta}(s, a)$ = probability of taking action a in state s . This allows stochastic policies. The *policy gradient theorem* links the gradient of the loss function to the *score function* of the policy, $\nabla \log \pi$:

$$\nabla J(\theta) = E_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(s, a) q^{\pi_{\theta}}(s, a)].$$

For instance, for a softmax,

$$\begin{aligned}\pi_{\theta}(s, a) &\propto \exp(\phi(s, a)' \theta) \\ \nabla \log \pi &= \phi(s, a) - E_{\pi_{\theta}}[\phi(s, \cdot)]\end{aligned}$$

and for a Gaussian policy

$$\begin{aligned}a &\sim N(\mu(s), \sigma^2), \quad \mu(s) = \phi(s)' \theta \\ \nabla_{\theta} \log \pi &= \frac{(a - \mu(s)) \phi(s)}{\sigma^2}.\end{aligned}$$

The **Monte Carlo policy gradient** (“REINFORCE”) algorithm estimates this expectation using one MC sample, i.e., $q^{\pi_{\theta}}(s, a) \approx r_t$.

The **actor-critic** method learns both the action-value function and the policy (it may be biased, but reduces

the variance)

$$\begin{aligned}q_w(s, a) &\quad \text{critic} \\ \pi_{\theta}(s, a) &\quad \text{actor} \\ \Delta w &= \text{as above} \\ \Delta \theta &= \alpha \nabla_{\theta} \log \pi_{\theta}(s, a) q_w(s, a).\end{aligned}$$

The **advantage function** critic replaces $q_v(s, a)$ with the advantage function

$$A^{\pi_{\theta}}(s, a) = q^{\pi_{\theta}}(s, a) - v^{\pi_{\theta}}(s)$$

(this does not change the expectation but reduces the variance). The **TD error** is an unbiased estimate of the advantage function

$$\delta = r + \gamma V_v(s') - V_v(s)$$

that only requires an estimate of v instead of q . (As before, you can replace the TD(0) target $r + \gamma v(s')$ with the return (MC target), the λ -return (TD(λ) target) or use eligibility traces (backward-view TD(λ)).)

8. Model-based reinforcement learning learns the model (an MDP: the state and action spaces are known, the reward $(s, a) \mapsto r$ is a regression problem, the transitions $(s, a) \mapsto s'$ a density estimation problem, possibly with a Bayesian prior) and uses it to estimate the value function or policy.

Dyna learns the model from experience and the value function from both real experience and simulated experience.

Forward search focuses on the sub-MDP starting at the current state. In particular, **Monte Carlo tree search** (MCTS) applies MC control to simulated experience (at first, we do not know the $Q(s, a)$, so we act at random until the end; this gives an estimate of some of the $Q(s, a)$; iterate with more and more paths – this is how computers play go).

TD search applies SARSA to the sub-MDP starting at the current state.

Dyna2 uses two sets of weights, long-term (TD learning, from real experience) and short-term (TD search), and adds them.

9. A **multi-arm bandit** is a 1-state, 1-step MDP.

action value	$q(a) = E[R \mid A = a]$
optimal value	$v^* = \text{Max}_a q(a)$ (unknown)
regret	$\ell_t = E[v^* - q(A_t)]$
total regret	$L_t = E\left[\sum_{\tau \leq t} v^* - q(A_\tau)\right]$
gap	$\Delta_a = v^* - q(a)$ (unknown)

We want to play the arms (actions) with a low gap more often. For the *greedy* (no exploration) and ε -*greedy* strategies, the loss L_t is linear in t . *Optimistic initialization* initializes $q(a)$ to a high value; it is still linear. *Decaying ε -greedy* is sublinear, but choosing the schedule requires knowledge of the gaps.

“Optimism in the face of uncertainty” uses an upper bound on $q(a)$: $q(a) \leq \hat{q}_t(a) + \hat{u}_t(a)$ with high probability. For instance, **UCB1** uses *Hoeffding’s inequality* (which assumes q bounded):

$$a_t = \underset{a}{\text{Argmax}} \hat{q}_t(a) + \sqrt{\frac{2 \log t}{N_t(a)}}.$$

Bayesian bandits put a prior on q and use some quantile of the posterior, e.g., $\hat{q}(a) + c\sigma(a)$.

Probability matching tries to estimate

$$P[\forall a' q(a) > q(a')]$$

from the posterior. **Thompson sampling** crudely estimates this probability by sampling from $q(a)$ (once for each action) and choosing the best action.

Information state search extends the state space to include all the information so far: the bandit becomes an MDP (with an infinite state space). For instance, a Bernoulli bandit gives an infinite tree of Beta distributions, and the values can be computed exactly (Gittins index) or with MCTS.

A **contextual bandit** is a 1-step MDP (with several states).

For MDPs, UCB is trickier: try optimistic initialization or the information state approach (with MCTS).

10. Let us consider 2-player zero-sum, perfect information (“Markov”) games. Nash equilibria (*joint* strategies) exist (provided you allow mixed strategies) but need not be unique. **Minimax** (depth-first) search

$$\begin{aligned} \pi &= \langle \pi_1, \pi_2 \rangle \\ v_\pi(s) &= E_\pi[G_t \mid S_t = s] \\ v_*(s) &= \text{Max}_{\pi_1} \text{Min}_{\pi_2} v_\pi(s) \end{aligned}$$

explores the tree of possible actions – usually truncated, with the leaf values estimated with some evaluation function.

Simple TD learns the value function with TD and uses it with minimax search. **TD root** uses the minimax value (of the root of the tree) as the target. **TD**

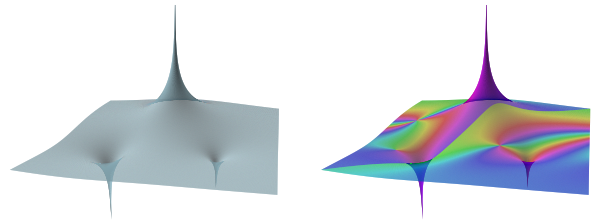
leaf does not update the root but the leaf of the search tree (run a search from s_t , take a decision, end up in s_{t+1} , run a search from s_{t+1} , update the leaf of the first search towards the value of the second search). **Treestrap** updates all the values in the search (not just the root or the leaf), *i.e.*, updates a shallow search from a deep one. MCTS (simulate games until the end, from the current state, and apply RL to those self-play games) works well. UCB can be generalized (UCT).

Phase plots of complex functions: a journey in illustration

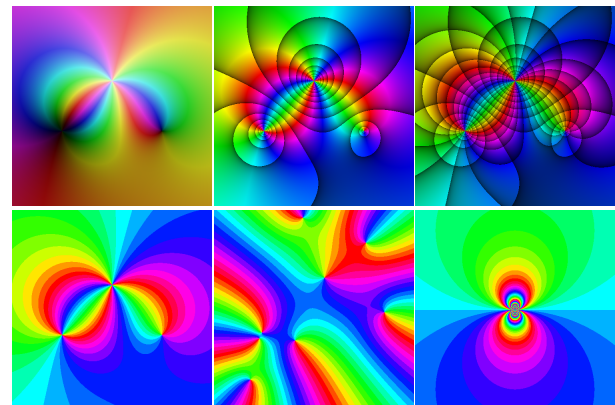
E. Wegert and G. Semmler (2011)

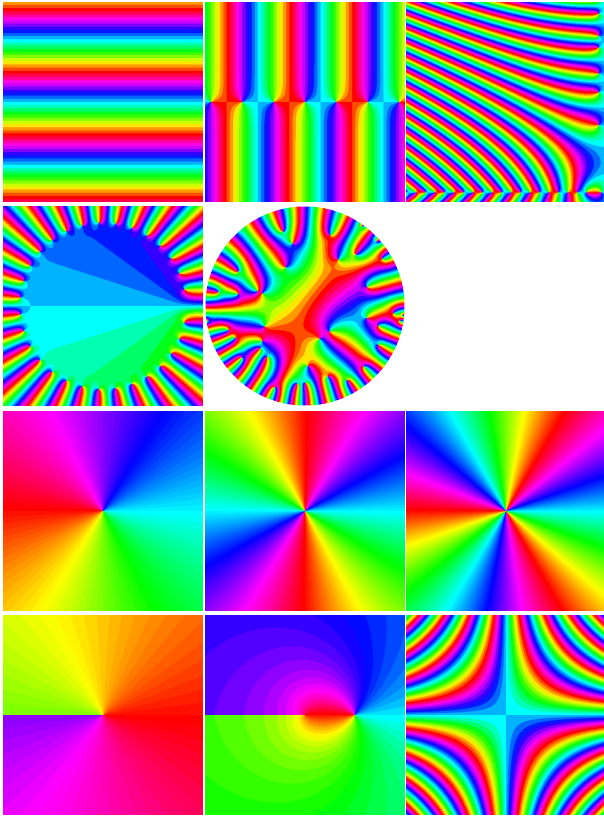
To plot a function f of a complex variable, one often considers the surface $z \mapsto |f(z)|$ (or $\log |f(z)|$), sometimes with a colour for the phase $f(z)/|f(z)| \in \mathbf{S}^1$. *Phase plots* only plot the phase, as a colour in the plane (one can also change the brightness, using a sawtooth function of $\log |f(z)|$ or $f(z)/|f(z)|$ or a product of them). Most of the information is still visible, sometimes more clearly than with the surface:

- Zeroes and poles are clearly visible: the colours changes on a simple loop give the difference between the number of zeroes and poles (counted with multiplicity) inside the loop;
- The zeroes of f' (that are not zeroes of f) are “colour saddle points”, *i.e.*, intersections of isochromatic lines;
- Essential singularities are obvious;
- Some results have a striking illustration, e.g., the zeroes of the partial sums of a converging power series with finite convergence radius R cluster at every point on $|z| = R$.



```
## Error in loadNamespace(x): there is no
package called 'elliptic'
```





**Introducing *elliptic*, an R package
for elliptic and modular functions**

R.K.S. Hankin

The **elliptic** R package provides tools to deal with (and plot) complex functions, in particular elliptic functions (doubly periodic meromorphic functions) such as Weierstrass's \wp and related $(\zeta' = -\wp, \sigma'/\sigma = \zeta)$ functions.

**Topology and data
G. Carlsson (2009)**

1. Topological data analysis estimates the **Betti numbers** $\beta_k(X, F) = \dim H_k(X, F)$ of a topological (or metric) space X (often, a subspace of \mathbf{R}^n), over a field F (often \mathbf{F}_2), from a finite number of points sampled from X .

An abstract *simplicial complex* is a pair (V, Σ) , where V is a finite set and $\Sigma \subset \mathcal{P}(V)$ such that $\sigma \in \Sigma, \tau \subset \sigma \implies \tau \in \Sigma$. Its topological realization is

$$\bigcup_{\sigma \in \Sigma} \text{chull}(\{e_i : i \in \sigma\})$$

where $(e_i)_{i \in V}$ is the canonical basis of \mathbf{R}^V .

The *homology* of a simplicial complex $X = (V, \Sigma)$ is

defined as

$$\Sigma_k = \{\sigma \in \Sigma : \#\sigma = k + 1\} \quad k\text{-simplices}$$

$$C_k = \mathbf{Z}^{\Sigma_k} \quad k\text{-chains}$$

$$(V, \leq) \quad \text{total order}$$

$$d_i(\sigma) = \sigma \setminus i\text{th element of } \sigma$$

$$d_i : \Sigma_k \longrightarrow \Sigma_{k-1}$$

$$\partial_k = \sum (-1)^i d_i : C_k \longrightarrow C_k$$

$$\partial_k \circ \partial_{k+1} = 0, \text{ therefore } \text{Im } \partial_{k+1} \subset \ker \partial_k$$

$$H_k(X, \mathbf{Z}) = \frac{\ker \partial_k}{\text{Im } \partial_{k+1}}.$$

Let $\mathcal{U} = (U_\alpha)_{\alpha \in A}$ be an open covering of a topological space X . The **nerve** $N\mathcal{U}$ is the abstract simplicial complex (A, Σ) where $\sigma = \{\alpha_0, \dots, \alpha_k\} \in \Sigma$ iff $\bigcap_{\alpha \in \sigma} U_\alpha \neq \emptyset$. If all intersections of elements of \mathcal{U} are empty or contractible, it is homotopy equivalent to X .

The **Čech complex** $\check{C}(V, \varepsilon)$ is the nerve of the covering of X with balls of radius ε with centers in a finite subset $V \subset X$. If X is compact, Riemannian, ε sufficiently small and V well chosen, then $\check{C}(V, \varepsilon)$ is homotopy equivalent to X .

The **Vietoris-Rips complex** $\text{VR}(V, \varepsilon)$ is (V, Σ) , with $\sigma = \{x_0, \dots, x_k\} \in \Sigma$ iff $\forall i, j, d(x_i, x_j) \leq \varepsilon$, i.e., all (x_i, x_j) span a Čech 1-simplex. (Its 1-skeleton is the “truncated graph” from the distance matrix.)

The **Delaunay complex** is the nerve of the covering by Voronoi cells

$$V_\lambda = \{x \in X : x \text{ closer to } \lambda \text{ than to any other } \lambda' \in V\}.$$

The **strong witness complex** of X , wrt a finite set of points (landmarks) $\mathcal{L} \subset X$, is a fattening of the Delaunay complex, $W^s(X, \mathcal{L}, \varepsilon) = (\mathcal{L}, \Sigma)$ with $\{\ell_0, \dots, \ell_k\} \in \Sigma$ iff

$$\exists x \in X : \forall i, d(x, \ell_i) \leq d(x, \mathcal{L}) + \varepsilon.$$

The **weak witness complex** $W^w(X, \mathcal{L}, \varepsilon)$ is (\mathcal{L}, Σ) with $\{\ell_0, \dots, \ell_k\} \in \Sigma$ iff

$$\forall \Lambda \subset \{\ell_0, \dots, \ell_k\} \quad \exists x \in X \quad \forall \ell \in \mathcal{L} \setminus \Lambda \quad \forall \ell_i \in \Lambda \\ d(x, \ell) + \varepsilon \geq d(x, \ell_i).$$

There are Vietoris-Rips variants W_{VR}^s and W_{VR}^w .

All those complexes are *functorial* in ε : the inequality $\varepsilon \leq \varepsilon'$ induces an inclusion $\check{C}(V, \varepsilon) \hookrightarrow \check{C}(V, \varepsilon')$, which in turn induces a morphism of homology groups $H_k \check{C}(V, \varepsilon) \longrightarrow H_k \check{C}(V, \varepsilon')$.

The **R-persistence homology** is the functor

$$\left\{ \begin{array}{ccc} (\mathbf{R}_+^{>0}, \leq) & \longrightarrow & \mathfrak{Mod} \mathbf{Z} \\ \varepsilon & \longmapsto & H_k \check{C}(V, \varepsilon). \end{array} \right.$$

R-persistence \mathbf{Z} -modules are complicated objects, but there is a classification theorem for finitely-generated **N-persistence** F -vector spaces,

$$(V_n)_{n \in \mathbf{N}} \equiv \bigoplus_{i=0}^N U(m_i, n_i),$$

where

$$U(m, n)_t = \begin{cases} F & \text{if } m \leq y \leq n \\ 0 & \text{otherwise} \end{cases}$$

and $U(m, n)_s \rightarrow U(m, n)_t$ is the identity for $m \leq s \leq t \leq n$. Note that $\text{---} \neq \text{---}$ even though all the dimensions are the same.

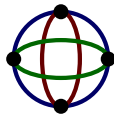
The \mathbf{N} -persistence F -homology of a finite set X (choose a subset $\{\varepsilon_n, n \in \mathbf{N}\} \subset \mathbf{R}$) can be represented as a **barcode**.

The actual computations require linear algebra (diagonalizing ∂ – Smith normal form) over the principal ideal domain $F[t]$ (\mathbf{N} -persistence vector spaces are equivalent to graded $F[t]$ -modules).

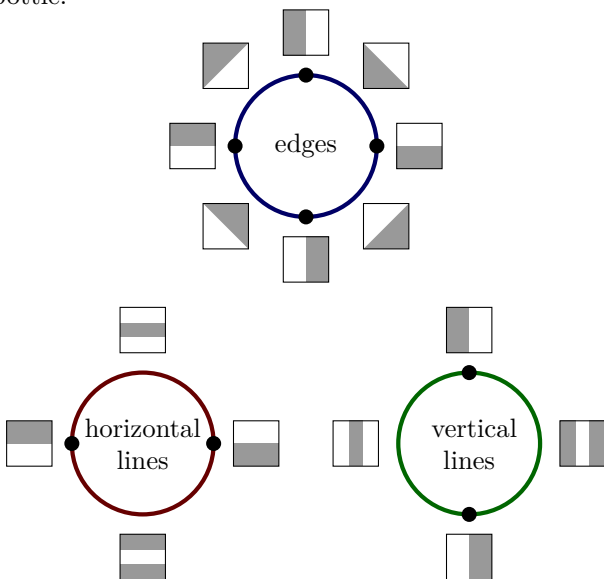
2. Those ideas have been applied to image data:

- Consider 3×3 patches (points in \mathbf{R}^9); only keep those with high contrast (top 20%);
- Center and normalize (points in \mathbf{S}^7); sub-sample;
- Estimate the density with the distance to the nearest neighbours ($k = 15$) and keep the top $T\%$ densest points ($T = 20\%$);
- Build the witness complex on 50 landmark points chosen by **archetypal analysis**.

The barcodes suggest $\beta_0 = 1$, $\beta_1 = 5$, which can be realized as the union of 3 circles,



corresponding to edges (or gradients), horizontal lines and vertical lines (consistent with the presence of horizontal and vertical artefacts: horizon, buildings, people, trees, etc.). Looking at β_2 suggests (unconvincingly) that those circles may be embedded in a Klein bottle.



For datasets in which one observation is a cloud of points (e.g., a few statistics estimated on a moving window – say, the number of spikes for the 5 neurons firing the most, in brain activity data), you can use

Betti numbers (or the most frequent Betti signatures) as features.

3. Given a map $\rho : X \rightarrow Z$ and a covering \mathcal{U} of Z (with $Z = \mathbf{R}$ or \mathbf{R}^n or \mathbf{S}^1), $\rho^*\mathcal{U}$ is a covering of X ; consider the Čech complex of its connected components $\check{C}^{\pi_0}(\rho^*\mathcal{U})$, e.g., for $\mathcal{U} = \mathcal{U}[R, e] = \{[kR - e, (k_1)R + e], k \in \mathbf{Z}\}$. For point clouds, replace the connected components π_0 with single linkage clustering: points less than ε apart are in the same component (these are the Vietoris-Rips connected components). The map ρ could be a density estimator, data depth

$$\rho_p(x) = \frac{1}{|X|} \sum_{y \in X} d(x, y)^p$$

or eigenvectors of the graph Laplacian of the 1-skeleton of the Vietoris-Rips complex, seen as (eigen) functions. To avoid choosing ε , consider the simplicial complex $SS(X, \rho, \mathcal{U})$ whose vertices are (α, I) , with $\alpha \in A$ and $I = (\varepsilon_i, \varepsilon_{i+1})$ a (maximal) interval on which the barcode of $H_0\check{C}(\rho^{-1}U_\alpha, \varepsilon)_\varepsilon$ does not change and whose k -simplices are the $\{(\alpha_0, I_0), \dots, (\alpha_k, I_k)\}$ such that $A_{\alpha_0} \cap \dots \cap A_{\alpha_k} \neq \emptyset$ and $I_0 \cap \dots \cap I_k \neq \emptyset$. Choose a section of the projection, $s : \check{C}\mathcal{U} \rightarrow SS$, $s(\alpha) = (\alpha, I_\alpha)$, and choose $\varepsilon_\alpha \in I_\alpha$. [The results are similar to Isomap. I am not convinced this is significantly different from a minimum spanning tree.]

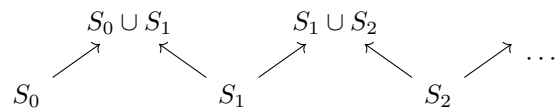
4. Persistent homology can be generalized to other index posets. For instance, $\text{VR}(X[T], \varepsilon)_{\varepsilon, T}$ where $X[T] = \{x \in X : \phi(x) \geq T\}$ and ϕ is a density estimator on the finite set X (or some other function – this is *Morse theory*), gives $\mathbf{R} \times \mathbf{R}$ - (or $\mathbf{N} \times \mathbf{N}$ -) persistent homology. It can be described with multigraded modules over $k[X_1, \dots, X_n]$ (here, $n = 2$). The classification of those modules is too complex to be useful (but may be amenable to *Gröbner bases*), but simpler invariants are more accessible, e.g., $\dim M_{t_1, \dots, t_n}$ or

$$\text{rank}(M_{t_1, \dots, t_n} \rightarrow M_{t'_1, \dots, t'_n})$$

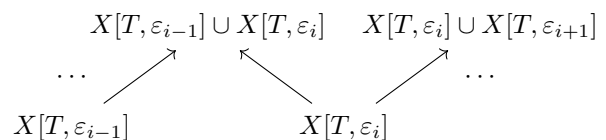
(for $n = 1$, the ranks contain all the information).

Zigzag diagrams (*quiver representations*) appear in the following situations:

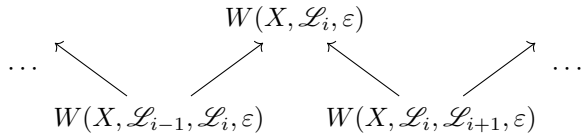
- Pairwise comparisons of subsamples (bootstrap)



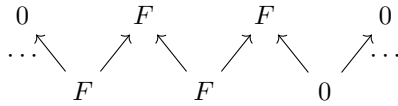
- Samples from a moving window;
- Upper level sets ($T\%$) of a density estimators with varying bandwidths ε_i



- Witness complexes as the set of landmarks change



There is a classification theory for finite zigzag persistence vector spaces: the elementary blocks are of the form



5. The paper ends with a discussion of functorial clustering algorithms (functors from the category of finite metric spaces with isometries, embeddings, non-increasing embeddings, non-increasing maps, to the category of sets.)

Barcodes: the persistent topology of data
R. Ghrist (2008)

Another (shorter) clear review article.

Computational topology for point data: Betti numbers of α -shapes
V. Robins (2002)

To estimate the Betti numbers (intuitively, $\beta_k(X)$ is the number of k -dimensional holes in X) of $X \subset \mathbf{R}^d$ from a finite set of points $S \subset X$, attach spheres of increasing radius α at each point:

$$S_\alpha = \bigcup_{x \in X} B(x, \alpha)$$

$$\beta_k^\alpha(S) = \dim H_k(S_\alpha)$$

for $\alpha \geq d_{\text{Hausdorff}}(S, X)$. The persistent Betti number $\beta_k^{\alpha, \beta}(S) = \text{rank}(i_* : H_k(S_\alpha) \rightarrow H_k(S_\beta))$, for $\alpha \leq \beta$ and $i : S_\alpha \hookrightarrow S_\beta$, is, intuitively, the number of k -dimensional holes in S_α that are not filled in S_β .

The number of connected components $\beta_0^\alpha(S)$ can be obtained from the minimum spanning tree of S . Higher Betti numbers can be computed from the Delaunay complex of S_α in S (the dual of the Voronoi complex of S_α in S).

JavaPlex tutorial
H. Adams and A. Tausz (2015)

JavaPlex is a Java/Matlab library to compute the persistent homology of filtered simplicial complexes. The tutorial ends with the 3-circle structure of image patches.

A roadmap for the computation of persistent homology
N. Otter et al. (2015)

Comparison of software to compute persistent homology: JavaPlex (well documented), Dionysus (C++/Python, documented), Dipha (C++, MPI, command line interface, fast), Gudhi (C++, fast), Perseus, Simpers, jHoles, pHat, CHomP.

Three examples of applied and computational homology
R. Ghrist (2008)

Concrete applications of algebraic topology include:

- *Euler characteristic integration* (constructible sheaves) to aggregate sensor data: the Euler characteristic is a measure since, under some circumstances, $\chi(A \cup B) = \chi(A) + \chi(B) - \chi(A \cap B)$;
- *Persistent homology* to understand the shape of and identify the structures in high-dimensional datasets, e.g., natural images or brain activity;
- *Conley index theory* (a generalization of Morse theory) to distinguish between chaotic and noisy (experimental or simulated) data.

How to write a 21st century proof
L. Lamport (2012)

Advocacy for the replacement of “informal proofs” (which we have been writing since the 17th century) with “structured proofs”: sequences of statements, each with a proof (itself structured, if needed). It is still too early for mathematicians to write formal (computer-verified) proofs, but familiarity with them can be helpful; the appendix shows a proof in TLA^+ .

An extensible SAT-solver
N. Eén and N. Sörensson (2004)

Walk through the MiniSAT code – should you want to implement your own SAT solver

Conflict-driven clause learning SAT solvers
J. Marques-Silva et al. (2008)

DPLL is the basic SAT solving algorithm:

- Write the formula as $\bigwedge_i \bigvee_j x_{ij}$;
- If one of the clauses has only one term, set it to true;
- If one of the variables always appears with the same sign, e.g., always x , or always $\neg x$, set it to true (or false);
- When you can no longer apply the previous steps, choose a value for one of the variables; backtrack when needed.

But this is suboptimal: when we reach a contradiction, we backtrack and try another value for some variable, but if it is unrelated to the contradiction, nothing will change, and we will explore the same tree again and again. *Clause learning* identifies the cause of the contradiction and adds it as a new clause, to help prune the search tree.

To find one (or several) clauses to add, represent the current state of the exploration as a graph, whose nodes are the possible values of the variables and with the clauses used for the deductions as arrows (all inbound edges have the same label), and find a cut between the latest decision variable and the conflict node.

Practical applications of Boolean satisfiability

J. Marques-Silva

SAT applications include:

- Hardware verification: checking that two circuits are equivalent;
- Circuit testing: automatic test generation, for boolean circuits, assuming the failure is that of a single value stuck at 0 or 1;
- Planning: checking if a state is reachable in k steps, in a deterministic transition system, with several transition functions, corresponding to the decisions; temporal logic;
- Bioinformatics, e.g., inferring haplotypes from genotypes: a genotype is a string over the alphabet $\{0, 1, 2\}$ (wild, mutant, heterozygous); a haplotype is a string over $\{0, 1\}$; each genotype comes from two haplotypes; the genotypes are known but the haplotypes are not; we want the minimum number of haplotypes to explain the genotypes.

Bayesian structural time series models

S.L. Scott (2015)

Structural models (level, trend and seasonality) can be put in state-space form and used for ad campaign effectiveness measurement. The model can be extended to allow non-Gaussian innovations (by expressing them as an (infinite) mixture of Gaussians) and long-term trends (by using a mean-reverting AR(1) slope instead of a random walk).

In R, check the `bsts` and `CausalImpact` packages.

Experiments in the internet age: a modern look at the multi-armed bandit

S.L. Scott (2014)

Planning an experiment and running it until the end is suboptimal (in terms of regret): you spend a lot of time estimating precisely options that are suboptimal. In the 2-arm bandit setup, $P(\theta_1 > \theta_2)$ can be estimated analytically, or estimated by sampling from the posteriors of θ_1 and θ_2 : *Thompson sampling* uses a single sample and works well. Full factorial experiments are rarely possible, but one can design fractional factorial experiments to estimate the coefficients as precisely as possible [this is not unlike Bayesian optimization].

Big data, statistics and the internet

S.L. Scott (2014)

Gibbs sampling (sample from $\beta|\mu, V$; sample from $\mu, V|\beta$; iterate) cannot be directly, efficiently implemented with MapReduce (there is too much communication and virtually no CPU usage – the complexity

of most MapReduce algorithms is the volume of communications, not the computations). Instead, **consensus Monte Carlo** gives different bits of data to various workers, which work independently (sample from $\beta|\mu, V, \text{data}_w$; sample from $\mu, V|\beta, \text{data}_w$; iterate) and the results are combined. Contrary to distributed optimization (ADMM) or horizontal strategies for stochastic optimization (progressive hedging), it is a simple consensus – there are no increasing penalties to force convergence.

This can be applied to hierarchical Bayesian Poisson regression (2×10^7 observations, 10^4 groups, 10 variables).

Introduction to RKHS and some simple kernel algorithms

A. Gretton (2015)

Given a kernel $k : X \times X \rightarrow \mathbf{R}$ on a finite set X , one can consider the Hilbert space

$$H = \text{Span}\{k(x, \cdot), x \in X\} \subset \mathcal{F}(X, \mathbf{R})$$

$$\left\langle \sum_x \alpha_x k(x, \cdot), \sum_y \beta_y k(y, \cdot) \right\rangle := \sum_{x,y} \alpha_x \beta_y k(x, y).$$

It satisfies:

$$H \subset \mathcal{F}(X, \mathbf{R})$$

$$\forall x \in X \quad k(x, \cdot) \in H$$

$$\forall x \in X \quad \forall f \in H \quad \langle f, k(x, \cdot) \rangle = f(x)$$

$$\|\delta_x\| = k(x, x)^{1/2} < \infty,$$

where

$$\delta_x : \begin{cases} H & \longrightarrow \mathbf{R} \\ f & \longmapsto f(x) \end{cases}$$

is the evaluation function.

Those results generalize to infinite X , and H is called a *reproducible kernel Hilbert space* (RKHS) for k .

Scalable Bayesian optimization using deep neural networks

J. Snoek et al. (2015)

Gaussian-process-based optimization scales cubically with the number of observations. Instead, one can train a neural net on the data

$$\text{input} \xrightarrow{\tanh} \dots \xrightarrow{\tanh} \phi_1 \dots \phi_D \xrightarrow{\text{linear}} \text{output}$$

(prefer tanh to ReLU) and fit a Bayesian linear model using the last layer $y = \sum \beta_i \phi_i + \varepsilon$ (*adaptive basis regression*).

Fast exact summation using small and large superaccumulators

R.M. Neal (2015)

Naive computation of long sums can lead to rounding error accumulation. There are approximate algorithms (Kahan-Babuška), but it is actually possible to compute those sums exactly, by using a “superaccumulator”, i.e., 4096 64-bit numbers, one for each possible

exponent (all but one bit overlap: carry propagation is rarely needed). For small sums, a smaller number of 64-bit numbers (e.g., with 32-bit overlaps) is sufficient. The overhead is twofold. [A big fixpoint number, with no overlap, looks simpler, but would force us to reimplement addition and could end up slower.]

Quantifying creativity in art networks
A. Elgammal and B. Saleh (2015)

To measure creativity (originality and influence):

- Build a graph of paintings, with a directed edge between two paintings if the first precedes the second, weighted by similarity;
- Subtract some reference value from the weights;
- Invert the negative edges;
- Compute the eigenvector centrality (PageRank).

An R package flare for high dimensional linear regression and precision matrix estimation
X. Li et al.

Variants of the lasso where the square loss is replaced with the absolute value, the L^2 norm (not its square), or more generally the L^p norm, $1 \leq p \leq 2$, and applications to sparse precision matrix estimation (TIGER, CLIME).

Tiger: a tuning-insensitive approach for optimally estimating Gaussian graphical models
H. Liu and L. Wang (2012)

Fitting a Gaussian graphical model, *i.e.*, estimating a sparse precision matrix, can be done column by column, with sparse regressions $X_i \sim X_{\setminus i}$: the lasso and many variants (Dantzig selector, scaled lasso, etc.) have been suggested, but the optimal tuning parameters cannot be computed in practice. The **sqrt lasso** (minimize $\|\text{residuals}\|_2 + \lambda \|\beta\|_1$ – this is the L^2 norm, not its square) is tuning-insensitive.

In R, check the **bigmatrix** and **flare** packages.

Identification of structured dynamical systems in tensor product reproducing kernel Hilbert spaces
M. Signoretto and J.A.K. Suykens

Factorization machines (or tensor machines) with a low rank constraint: add a “multilinear spectral penalty”, built from the SVDs of the unfoldings of the model parameters.

Improving distributional similarity with lessons learned from word embeddings
O. Levy et al.

Traditional and neural word embeddings use the same bag-of-contexts representation:

- Positive pointwise mutual information;

$$\text{PPMI}(\text{word}, \text{context}) = \left(\log \frac{\hat{P}(\text{word}, \text{context})}{\hat{P}(\text{word})\hat{P}(\text{context})} \right)_{+}$$

- Its SVD-based low-rank approximation;
- Skipgram with negative sampling (SGNS, wordvec): maximize $s(\text{word} \cdot \text{context})$ and minimize $s(\text{word} \cdot \text{corrupted context})$, where s is the sigmoid function and \cdot the scalar product – this can be seen as a matrix factorization $\text{PMI} \approx \text{Words} \cdot \text{Contexts}' + \log k$, with a robust (sigmoid) loss;
- Global vectors (GloVe), a factorization $M \approx \text{Words} \cdot \text{Contexts}' + b_w \cdot \mathbf{1}' + \mathbf{1} \cdot b_c'$, where $M_{w,c} = \log \#(w, c)$.

The difference in performance is mostly due to better (or learned) hyperparameters in the “neural” networks.

Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms
C. Thornton et al.

Use Bayesian optimization, e.g., SMAC (random-forest-based Bayesian optimization) or TPE (tree-structured Parzen estimators), to choose both algorithm and hyperparameters in Weka.

Real-time prediction and post-mortem analysis of the Shanghai 2015 stock market bubble
D. Sornette et al. (2015)

- Fit the log-periodic power law (LPPL) model on several windows, from 750 to 125 trading days;
- Filter the results (they give a long list of empirical conditions the model should satisfy);
- The “confidence” is the proportion of models (*i.e.*, window sizes) that pass those filters; it estimates crash risk;
- The “trust” is a resampling-based variant of that goodness-of-fit test: resample the residuals, add them to the fit, re-estimate the model, check if it still passes the filters;
- Compute the probability distribution function of the crash time, or of the bubble start time.

Generative adversarial nets
I.J. Goodfellow et al.

Generative adversarial nets simultaneously learn two models

generator G : noise \mapsto data
discriminator D : noise or data \mapsto true or false

via the minimax game with value

$$V(D, G) = E_{x \sim \text{data}} \log D(x) + E_{x \sim \text{noise}} \log(1 - DG(x))$$

$$\text{value}(\text{game}) = \min_G \max_D V(D, G).$$

Conditional generative adversarial nets

M. Mirza and S. Osindero

GANs can be generalized to conditional models (models of $x|y$ instead of x).

Deep generative image models using a Laplacian pyramid of adversarial networks

E. Denton et al.

Stack GANs to generate finer and finer images.

A neural conversational model

O. Vinyals and Q.V. Le

Build a conversational model with a sequence-to-sequence (seq2seq) RNN-LSTM network, trained on IT helpdesk chats or movie subtitles.

Learning to understand phrases by embedding the dictionary

F. Hill et al.

Train a RNN with LSTM to map dictionary definitions to word vector representations (word2vec) and use as a crossword solver (or a reverse dictionary, *i.e.*, to find the word on the tip of your tongue from its definition).

A simple way to initialize recurrent networks of rectified linear units

Q.V. Le et al.

To avoid the vanishing/exploding gradient problems when training recurrent neural networks (RNN) with stochastic gradient descent (SGD):

- Use Hessian-free optimization instead of SGD;
- Or use SGD, with momentum, careful initialization, and clipped gradients;
- Or use a LSTM network (the model is complicated, but its gradients well-behaved);
- Or use ReLU and careful initialization.

Towards large-scale continuous EDA: a random matrix theory perspective

A. Kabán et al.

In high dimension, one can still use CMA-ES after imposing some structure on the covariance matrix (e.g., diagonal). Instead, one can apply CMA-ES on random (low-dimensional) subspaces and combine the resulting subpopulations. (EDA, “estimation of distribution algorithms”, refers to CMA-ES-like evolutionary algorithms.)

Centrality of the supply chain

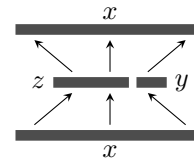
L. Wu

Supplier and consumer centralities (HITS algorithm).

Discovering hidden factors of variation in deep networks

B. Cheung et al.

Train a semi-supervised auto-encoder



(we want a latent representation (z, y) of x , where y is known, and z should be as independent of y as possible) with a penalty on the average squared covariance $\langle \text{Cov}(z_i, y_i)^2 \rangle$.

A MiniZinc tutorial

K. Marriott and P.J. Stuckey

MiniZinc is a discrete optimization modeling language: it translates the problem to feed it to various optimizers (constraint propagation, linear programming, etc.) and provides global constraints (alldifferent, etc.).

Modeling discrete optimization

P.J. Stuckey and C. Coffrin (Coursera, 2015)

Another MiniZinc tutorial, with exercises.

The Wiener-Askey polynomial chaos for stochastic differential equations

D. Xiu and G.E. Karniadakis (2001)

The **polynomial chaos** expansion of a stochastic process (with finite variance) is the approximation

$$X_t = a_0(t)H_0 + \sum_{i_1 \geq 0} a_{i_1}(t)H_1(X_{i_1}) + \sum_{i_1, i_2 \geq 0} a_{i_1 i_2}(t)H_2(X_{i_1}, X_{i_2}) + \dots$$

where X_1, X_2, \dots are $N(0, 1)$ iid,

$$H_n(x_1, \dots, x_n) = e^{-\frac{1}{2}x'x} (-1)^n \frac{\partial^n}{\partial x_1 \dots \partial x_n} e^{-\frac{1}{2}x'x}$$

are Hermite polynomials and a_{i_1, \dots, i_k} are functions to be determined. It can be generalized to other families of orthogonal polynomials and non-Gaussian random variables. These expansions can be used to solve SDEs, in the same way one would use power series with ODEs.

Time-dependent polynomial chaos

P. Vos

Worked out examples showing how to numerically solve stochastic ODEs (ODEs with random variables in their

coefficients or their initial conditions) with generalized polynomial chaos (gPC) expansions and to circumvent numeric instability (by reinitializing the expansion from time to time).

Polynomial chaos: a tutorial and critique from a statistician's perspective
A. O'Hagan (2013)

Polynomial chaos (PC) describes a random variable X as $X \stackrel{d}{=} f(\Xi)$ (equality in distribution), where Ξ is a random variable following a known, simple distribution. Such a function f is not unique: one can look for it in the finite-dimensional space of polynomials (of degree at most n) orthogonal wrt the probability distribution function of Ξ (Legendre for $U(-1, 1)$, Hermite for $N(0, 1)$, Laguerre for $\text{Exp}(1)$ – choose depending on the boundedness of X). Typically, Ξ has dimension at least that of X . The Karhunen-Loève expansion is a degree-1 PC expansion.

Random geometry on the sphere
J.F. Le Gall (2014)

To obtain a random metric on the sphere, draw a random planar graph on the sphere, consider the graph distance (suitably rescaled), refine it by adding more nodes and edges until, in the limit, you have a metric on the whole sphere. More precisely, the “random graphs” are defined as follows:

- Consider *planar maps*, i.e., embeddings of graphs into the sphere \mathbf{S}^2 , up to homeomorphism;
- Consider only p -angulations (i.e., the faces have p edges);
- Sample uniformly from

$$M_n^p = \{p\text{-angulations with } n \text{ faces}\},$$

for $p = 3$ or 4 and n large.

For the limit, notice that a planar map is an element of

$$K = \{\text{compact metric spaces}\}/\text{isometries},$$

which is complete when equipped with the Gromov-Hausdorff distance

$$\begin{aligned} d(E_1, E_2) &= \inf\{d_H(\psi_1 E_1, \psi_2 E_2) : \\ &\quad \psi_i : E_i \hookrightarrow E \text{ isometric embeddings}\} \\ d_H(X, Y) &= \text{Max}\left\{\sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y)\right\} \\ &= \inf\{\varepsilon \geq 0 : X \subset Y_\varepsilon \text{ and } Y \subset X_\varepsilon\} \end{aligned}$$

where X_ε is the ε -enlargement of X .

The **Brownian map** is the random variable, with values in K , obtained in the limit; it is a sphere, but it has Hausdorff dimension 4 (a.s.).

Quadrangulations can be described as *well-labeled trees* (Schaeffer's bijection).

A **continuous random tree** (CRT) is obtained from a *Brownian excursion* (a Brownian motion B such that

$B_0 = B_1 = 0$ and $\forall t \ B_t \geq 0$), as $[0, 1]/\sim$ where $s \sim t$ if $d(s, t) = 0$ and

$$d(s, t) = B_s + B_t - 2 \min_{u \in [s, t]} B_u;$$

is a distance on the tree. Schaeffer's construction can be applied to a CRT with Brownian labels (there are two levels of randomness: in the tree, and in the labels).

Aspects of random maps
G. Miermont (2014)

All the details on the Brownian map.

A relation between Brownian bridge and Brownian excursion
W. Vervaat (1979)

A **Brownian excursion** is a Brownian motion E on $[0, 1]$ conditioned by $E_0 = E_1 = 0$ and $\forall s \in [0, 1], E_s \geq 0$. It can be built from a Brownian bridge by finding its minimum and shifting it (modulo 1) to start and end at that minimum:

$$\begin{aligned} W_t & \quad \text{Brownian motion} \\ B_t = W_t - tW_1 & \quad \text{Brownian bridge} \\ \tau = \underset{t}{\text{Argmin}} B_t & \\ E_t = B_{\tau+t \bmod 1} - B_\tau & \quad \text{Brownian excursion.} \end{aligned}$$

Automatic construction and natural language description of nonparametric regression models
J.R. Lloyd et al.

The **automatic statistician** models time series as Gaussian processes (GP) whose kernel is described via a formal language, using elements such as WN (white noise), C (constant), Lin (linear), SE (square exponential), Per (periodic), CP (change points) and $+$ and \times operators. The models are not unlike *symbolic regression* and include linear regression ($C + \text{Lin} + \text{WN}$), GP smoothing ($\text{SE} + \text{WN}$), cyclical decomposition ($\sum \text{SE} + \sum \text{Per} + \text{WN}$), Fourier decomposition ($C + \sum \cos + \text{WN}$), etc.

Intelligible models for classification and regression
Y. Lou et al. (2012)

Comparison of variants of GAM: the shape functions can be splines (in R: `mgcv`), regression trees, tree ensembles (bagged, boosted, or both) with a fixed or adaptive depth; the model can be learnt via penalized least squares (wiggleness penalty, $\int |f''|^2$), gradient boosting or backfitting (learn f_k on the residuals $y - \sum_{i \neq k} f_i(x_i)$, iterate until convergence): boosted bagged trees, preferably with adaptive depth, perform better.

***Accurate intelligible models
with pairwise interactions***
Y. Lou et al. (2013)

Interactions can easily be added to GAMs (GA²M), but since the number of pairs of variables can be huge, some form of feature selection is needed. For instance, one can greedily add the most promising interaction to the model, using

- Anova (p -value of a test comparing the model with no interactions and the model with the (x_i, x_j) interaction);
- Independence χ^2 test between the sign of the GAM residuals and $(x_i > a_i, x_j > a_j)$, $a_i = \text{median}(x_i)$;
- Comparison of the RMSE of a full model (e.g., a random forest) $y \sim \text{rf}(x_1, \dots, x_n)$ with a model without the (x_i, x_j) interaction, $y \sim \text{rf}(x_1, \dots, \hat{x}_i, \dots, x_n) + \text{rf}(x_1, \dots, \hat{x}_j, \dots, x_n)$;
- RSS of the best split for (x_i, x_j) – it can be computed efficiently.

Convexifying the set of matrices of bounded rank: applications to the quasiconvexification and convexification of the rank function
J.B. Hiriart-Urruty and H.Y. Le (2011)

The convex hull of

$$\{M : \text{rank } M \leq k \text{ and } \|M\| \leq 1\}$$

is

$$\{M : \|M\|_* \leq k \text{ and } \|M\| \leq 1\}$$

where $\|M\| = \sum_{x \neq 0} \|Mx\|_2 / \|x\|_2 = \sigma_1(M)$ is the spectral (operator) norm and $\|M\|_* = \sum_i \sigma_i(M)$ is the nuclear (trace) norm (those norms are dual).

This generalizes a similar result for the ℓ^0 pseudo-norm: the convex hull of $\{x : \|x\|_0 \leq k \text{ and } \|x\|_\infty \leq 1\}$ is $\{x : \|x\|_1 \leq k \text{ and } \|x\|_\infty \leq 1\}$ and $\|\cdot\|_1$ and $\|\cdot\|_\infty$ are dual.

A function f is **quasi-convex** if its sublevel sets $[f \leq \alpha]$ are convex. The *(quasi-)convex hull* of a function is the largest (quasi-)convex function minorizing it. The restricted rank is

$$\text{rank}_r : M \mapsto \begin{cases} \text{rank } M & \text{if } \|M\| \leq r \\ +\infty & \text{otherwise.} \end{cases}$$

Its quasi-convex hull is

$$M \mapsto \begin{cases} \lceil \frac{1}{r} \|M\|_* \rceil & \text{if } \|M\| \leq r \\ +\infty & \text{otherwise} \end{cases}$$

and its convex hull is

$$M \mapsto \begin{cases} \frac{1}{r} \|M\|_* & \text{if } \|M\| \leq r \\ +\infty & \text{otherwise.} \end{cases}$$

In other words, the spectral norm $\|\cdot\|_*$ (sum of the singular values) is a convex relaxation of the rank. For instance, the non-convex **robust PCA** problem

$$\begin{aligned} &\text{Find} && L, S, N \\ &\text{To minimize} && \text{rank } L + \lambda \|S\|_0 \\ &\text{Such that} && \|N\|_F \leq \varepsilon \\ &\text{and} && X = L + S + N \end{aligned}$$

can be relaxed to

$$\begin{aligned} &\text{Find} && L, S, N \\ &\text{To minimize} && \|L\| + * + \lambda \|S\|_1 \\ &\text{Such that} && \|N\|_F \leq \varepsilon \\ &\text{and} && X = L + S + N. \end{aligned}$$

Robust models often look for a decomposition

$$\begin{aligned} \text{data} &= \text{model} + \text{outliers} + \text{noise} \\ &= \text{low rank} + \text{sparse} + \text{small}. \end{aligned}$$

***Robust rotation synchronization
via low-rank and sparse matrix decomposition***
F. Arrigoni et al. (2015)

Robust PCA and matrix completion are both low-rank approximations: they can be combined.

Rotation matrices $R_1, \dots, R_n \in \text{SO}(3)$ can be recovered from noisy relative alignments $R_{ij} = R_i R_j^{-1} + \text{noise}$ by minimizing

$$\sum_{R_{ij} \text{ available}} \|R_{ij} - R_i R_j^{-1}\|_F^2.$$

Letting $X = \begin{pmatrix} 1 & R_{12} & \cdots & R_{1n} \\ R_{21} & & & \vdots \\ \vdots & & & \\ R_{n1} & \cdots & & 1 \end{pmatrix}$, the problem becomes

$$\begin{aligned} &\text{Find} && R = \begin{pmatrix} R_1 \\ \vdots \\ R_n \end{pmatrix} \in \text{SO}(3)^n \\ &\text{To minimize} && \|p(X - RR')\|_F^2 \end{aligned}$$

where p is the projection on the available coordinates. Since $R \in \text{SO}(3)^n$, we can add the constraints $\text{rank } X \leq 3$ and $X \succcurlyeq 0$.

***Efficient computation of sparse Hessians
using coloring and automatic differentiation***
A.H. Gebremedhin et al. (2009)

To recover a sparse, symmetric matrix A , from its sparsity structure and a small number of products Ax_1, \dots, Ax_n , one can use a colouring of the graph whose adjacency matrix is the sparsity structure to define $S = (\mathbf{1}_{\text{node } i \text{ has colour } j})_{ij}$; for a *star-colouring* (each 4-vertex path has at least 3 colours; consequently, the subgraph induced by 2 colours is a collection of stars), one can efficiently recover A from AS .

***Compressed sensing recovery
via nonconvex shrinkage penalties***
J. Woodworth and R. Chartrand (2015)

Compressed sensing, *i.e.*, ℓ^0 minimization

$$\begin{array}{ll}\text{Find} & w \\ \text{To minimize} & \|w\|_0 \\ \text{Such that} & Aw = b\end{array}$$

is often relaxed to ℓ^1 minimization, and many algorithms rely on the *proximal mapping* of the ℓ^1 norm

$$S(x) = \text{Argmin}_w \lambda \|w\|_1 + \frac{1}{2} \|w - x\|_2^2.$$

The ℓ^p quasi-norms, $0 < p < 1$, are a better approximation of the ℓ^0 penalty, but their proximal mapping has no known closed form expression. Instead, one can start with a proximal mapping (in closed form), *e.g.*,



and derive the corresponding penalty .

Entropy-based financial asset pricing
M. Ormos and S. Zibriczky (2014)

In portfolio construction, one can use entropy instead of the standard deviation as a risk measure (it requires a density estimator): one can compute and plot efficient portfolios in the entropy \times return space and decompose the entropy into systematic (mutual information) and specific (conditional entropy) components.

***Permutation-information-theory approach
to unveil delay dynamics
from time series analysis***
L. Zunino et al. (2010)

Given a probability distribution p on $\llbracket 1, N \rrbracket$, one can define the normalized entropy H_S and the complexity C_{JS} as (here, p_e denotes the uniform distribution)

$$\begin{aligned} S[p] &= - \sum p_i \log p_i \\ S[p_e] &= \log N \\ H_S[p] &= \frac{S[p]}{S[p_e]} \\ J(p|p_e) &= S[\tfrac{1}{2}(p + p_e)] - \tfrac{1}{2}S[p] - \tfrac{1}{2}S[p_e] \\ \max_p J(p|p_e) &= \log 2N - (N + 1) \log(N + 1) \\ Q_J(p|p_e) &= \frac{J(p|p_e)}{\max_q J(q|p_e)} \\ C_{JS}[p] &= Q_J(p|p_e) H_S[p] \end{aligned}$$

To transform a time series into a discrete distribution, choose

- an embedding dimension $D > 1$;

- an embedding delay τ

and consider the permutation defined by

$$(x_t, x_{t+\tau}, \dots, x_{t+(D-1)\tau}).$$

This defines a discrete distribution on \mathfrak{S}_D (D should not be too large: $D! \ll n$.) The corresponding entropy and complexity are the *permutation entropy* and the *permutation statistical complexity*.

As an example, one can look at the Mackey-Glass oscillator

$$\begin{aligned} \dot{x} &= -x + \frac{ax(t - \tau_0)}{1 + x^c(t - \tau_0)} \\ a &= 2, \quad c = 10, \quad \tau_0 = 60 \end{aligned}$$

in the (H_S, C_{JS}) space, as τ_0 varies.

***Data manipulation detection
via permutation information theory quantifiers***
A.F. Bariviera (2015)

Application to financial time series.

***Representing numeric data in 32 bits
while preserving 64-bit precision***
R.M. Neal (2015)

Storing data as decimals (rather than floating point) is space-efficient, but a time-consuming conversion is usually needed to use the data. Instead, one can use a table lookup (implemented in pqR).

***A generalized
Kahan-Babuška-summation algorithm***
A. Klein (2005)

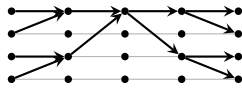
When naively computing a (long) sum of floating point numbers, the rounding errors accumulate. It is possible to compensate for them, *e.g.*, by re-ordering the terms (if they are all positive, use the Huffman code order, starting with the smallest numbers – unfortunately, with arbitrary signs, finding the optimal order is an NP-hard problem) or by estimating the error and correcting for it: the Kahan algorithm estimates the error and corrects it immediately; the more accurate Kahan-Babuška-Neumaier (KBN) algorithm corrects it at the end. There are dozens of variants of those algorithms.

***Printing floating-point numbers quickly
and accurately with integers***
F. Loitsch (2010)

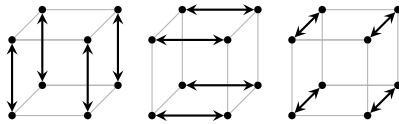
Printing floating-point numbers (*i.e.*, converting them to decimal) is not a trivial task. The previous algorithm, Dragon4, required arbitrary-precision arithmetic. Grisu, only requires integer arithmetic (but occasionally needs to fall back to Dragon4).

***Kylrix: a sparse allreduce
for commodity clusters***
H. Zhao and J. Canny

The all-reduce pattern usually involves a tree network



In a *bufferfly* network, the nodes are arranged in a hypercube; in step k , the nodes communicate with their neighbours along dimension k .



***Edge compression techniques
for visualization of dense directed graphs***
T. Dwyer et al. (2013)

To display a directed graph with a lot of edges (e.g., the dependencies between software components), one can try to group them into modules, with edges between modules instead of between nodes:

- One can find, in linear time (via hashing), nodes that have the same set of neighbours, and group them (the nodes inside a module are either disconnected or form a clique, depending on whether we include a node in its neighbourhood);
- One can also allow for some internal structure in the modules (but the algorithm is trickier);
- One can also allow for some module-crossing edges (powergraph), e.g., with a greedy algorithm: compute a hierarchical clustering of the nodes, for some node similarity measure; for each potential cluster, compute the number of edges that would be removed; choose the modules greedily. The results were compared with the optimal minimizer of

$$\#modules + w_1 \times \#edges + w_2 \times \#crossings,$$

computed using MiniZinc – the greedy solution is far from optimal but, in psychological tests (“how much of the graph do you remember?” “Can you find the shortest path from A to B ?”), it is an improvement on traditional layout algorithms.

***Improved optimal and approximate
power graph compression
for clearer visualization of dense graphs***
T. Dwyer et al. (2013)

Finding the optimal *powergraph compression* of a graph (i.e., a module decomposition that reduce the number of edges) using general optimization methods (integer propagation, integer programming) is too time-consuming.

The following heuristic is faster, and gives decent results:

- Arrange the set of possible solutions into a tree, with the initial graph (1-node modules) as the root, and children obtained by merging two modules in the parent;
- Use best-first-search or, better, *beam search*: at each iteration (i.e., for each depth in the tree), keep the k best solutions ($k = 1$ is greedy-best-first search).

It is actually reasonable to explore the whole tree, with backtracking, after judicious pruning.

***A generic algorithm
for layout of biological networks***
F, Schreiber et al. (2009)

The algorithm begins *dunnart* (for constrained graph layout): start with a feasible but partial layout, find a locally optimal partial layout, extend it into a full layout.

***Google matrix analysis
of the multiproduct world trade network***
L. Ermann and D.L. Shepelyansky (2015)

In the graph whose vertices are country \times product pairs, also consider the *Chei-rank* (the PageRank of the opposite graph); use a personalized variant of PageRank to account for imbalance between products (e.g., oil vs furs) or countries; and compute the correlations

$$\kappa = \sum_{c,p} p_{cp} p_{cp}^* - 1$$

$$\kappa_{p_1, p_2} = N_C \sum_c \frac{p_{cp_1} p_{cp_2}^*}{\sum_{c_1} p_{c_1 p_1} \sum_{c_2} p_{c_2 p_2}} - 1$$

where p_{cp} (resp. p_{cp}^*) are the PageRank (Chei-rank) scores (eigenvectors for $\lambda = 1$, normalized so that $p' \mathbf{1} = p'^* \mathbf{1} = 1$).

***Randomizing bipartite networks:
the case of the world trade web***
F. Saracco et al. (2015)

The common graph metrics can be generalized to bipartite graphs:

- Assortativity: $\text{Cor}(\deg X, \deg Y \mid \text{edge } X-Y)$;
- Complexity and fitness are PageRank analogues – they can be used to reorder the rows and columns of the adjacency matrix;
- As a replacement for the clustering coefficient (there are no odd cycles), one can count motifs, e.g., and
- Nestedness: count the number of products (countries) two countries (products) have in common; sum; normalize.

To construct statistical tests on a graph, use some maximal entropy distribution on the set of graphs (e.g., assuming the degree distribution is known).

***On the modular dynamics
of financial market networks***
F.N. Silva et al. (2015)

Let G be a graph (undirected, no self-loops, no multiple edges), $A(G)$ its adjacency matrix, $\Delta(G)$ its degree matrix (diagonal), $L(G) = \Delta(G) - A(G)$ its Laplacian, $\rho(G) = L(G)/\text{tr } \Delta(G)$ its density matrix, $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ the eigenvalues of $\rho(G)$. The **Von Neumann entropy** of G is

$$S(G) = - \sum_i \lambda_i \log_2 \lambda_i.$$

One can look at how graph metrics or community metrics (modularity, average shortest path length, average betweenness, degree assortativity, transitivity, von Neumann entropy) vary over time (for a network built from the correlation matrix of asset returns, estimated on a moving window).

With a model-based community detection algorithm, one can generate random graphs from the fitted community model, and look at the distribution of graph metrics.

***A note on the von Neumann entropy
of random graphs***
W. Du et al. (2010)

***Dynamic multi-factor clustering
of financial networks***
G.J. Ross (2015)

To measure how much a qualitative variable (sector, country, etc.) influences a hierarchical clustering (from stock return correlations): for each pair of points i, j in the same class, find their closest common ancestor k , count the proportion of points in this class among the descendants of k ; average over all pairs.

Identifying states in a financial market
M.C. Münnix et al. (2012)

To identify market states:

- Compute the correlation matrix, for daily (or hourly) returns, on a 2-month moving window;
- Compute the normalized L^1 distance (alternatively, the difference between the largest eigenvalues) between those correlation matrices, and cluster them (hierarchical clustering or k -means) – alternatively, fit some regime-switching model.

***Forecasting financial extremes: a network
degree measure of super-exponential growth***
W. Yan et al. (2015)

Use the degree of the *visibility graph* of the log-price (or its opposite) to identify super-exponential growth (or decay); this is similar to the LPPL pattern recognition indicator.

***A practical approach to financial crisis
indicator based on random matrices***
A. Kornprobst and R. Douady (2015)

The **Hellinger distance** between two probability distributions is

$$\text{Hellinger}(p, q) = \int (\sqrt{p} - \sqrt{q})^2.$$

To detect market instability, look at the following indicators:

- Hellinger distance between the Marcenko-Pastur distribution and the distribution of the eigenvalues of the correlation matrix (low eigenvalues are not informative: discard everything below the tenth of the maximum theoretical eigenvalue);
- Hellinger distance with the spectrum of the sample correlation of a constant correlation Gaussian (or Student, with 2 degrees of freedom);
- Maximum eigenvalue of $\text{Var } X$ (spectral radius);
- $\text{tr } \text{Var } X$;
- $\text{Max Spec Cor } X$, with capitalization-weighted and volume-weighted variants (apply a moving average filter if too noisy).

Kernel spectral clustering and applications
R. Langone et al. (2015)

To cluster data with PCA (or kernel PCA, or spectral methods – kPCA on a graph Laplacian), binarize the transformed data ($\text{sign PC}_1, \text{sign PC}_2, \dots$) and use the most frequent binary codes as clusters.

***Scale up nonlinear component analysis
with doubly stochastic gradients***
B. Xie et al.

Use two stochastic approximations simultaneously in kernel PCA: process the data in minibatches (as in stochastic gradient descent) and use random features (different in each batch, as in randomized PCA).

Implied correlation and expected returns
M. Valenzuela

The option-implied correlation

$$\rho = \frac{\sigma^2 - \sum_i w_i^2 \sigma_i^2}{\sum_{i \neq j} w_i w_j \sigma_i \sigma_j}$$

$$\sigma^2 = \frac{2e^{rT}}{F_T} \left(\int_0^{F_T} \text{put}_T(K) dK + \int_{F_T}^{\infty} \text{call}_T(K) dK \right)$$

computed from (30-day) put and call options on the S&P 100 and its constituents, can help predict (3- to 12-month) future returns.

Market timing with a robust moving average
V. Zakamulin (2015)

Among the half-dozen momentum strategies studies, the exponentially-weighted moving average (EWMA) “buy if $\sum_{k \geq 0} \lambda^k \times \text{returns}_{[t-k-1, t-k]} > 0$ ”, with $\lambda = 0.87$, is the most robust to the look-back period.

Market timing with moving averages: anatomy and performance of trading rules
V. Zakamulin

Technical analysis rules can be formulated as “buy if $\sum_{k \geq 0} w_k \text{Price}_{\text{now}-k}$ ” for different weighting schemes (one can also use price changes, with different weights).

Copula-based hierarchical risk aggregation
F. Derendinger (2015)

A *mildly tree-dependent* random vector $(X_i)_{i \in I}$ is

- a rooted tree structure on the index set I ,
- a univariate distribution for each leaf,
- a copula for each node, describing the dependence of its children,

where each node is the sum of its children. This does not uniquely determine the distribution of the random vector, unless one also assumes that, for each node, its descendants are conditionally independent from the other nodes.

Measuring financial asset return and volatility spillovers, with applications to global equity markets
F. Diebold and K. Yilmaz (2009)

To measure spillover in a VAR(p) model $x_t = \Phi x_{t-1} + \varepsilon_t$,

- Write the proces as an MA(∞) process, $x_t = (1 - \Phi L)^{-1} \varepsilon_t$;
- Using the Choleski decomposition $\text{Var } \varepsilon_t = Q_t Q_t^{-1}$, rewrite the process as an MA(∞) process with orthogonal innovations $x_t = (I - \Phi L)^{-1} Q_t^{-1} \cdot Q_t \varepsilon_t = A(L) u_t$;
- The 1-step-ahead error $x_{t+1} - x_{t+1|t} = A_0 u_{t+1}$ has variance $A_0 A_0'$; the K -step-ahead error can be decomposed

$$\sum_{ij} \sum_{1 \leq k \leq K} a_{ij}^{k,2} = \sum_i \left(\sum_k a_{ii}^{k,2} + \sum_{j \neq i} \sum_k a_{ij}^{k,2} \right)$$

where the second term is the spillover of j onto i .

The *spillover index* is

$$S = \frac{\sum_{i \neq j} \sum_k a_{ij}^{k,2}}{\sum_{ij} \sum_k a_{ij}^{k,2}}$$

It depends on the order of the variables, but not too much.

On the computational complexity of high-dimensional Bayesian variable selection
Y. Yang et al. (2015)

The Bayesian hierarchical sparse model

$$\begin{aligned} \gamma &= \{\text{variables entering the model}\} \subset \llbracket 1, n \rrbracket \\ \pi(\gamma) &\propto p^{-\kappa|\gamma|} \mathbf{1}_{|\gamma| \leq s_0} \\ \pi(\phi) &\propto \phi^{-1} \text{ (improper prior)} \\ \beta_\gamma &\sim N(0, g\phi^{-1} I_{|\gamma|}) \\ w &\sim N(0, \phi^{-1} I_n) \\ Y &= X + \gamma\beta_\gamma + w \end{aligned}$$

(the slab-and-spike prior, *i.e.*, a mixture of two Gaussians with different variances, is another popular choice), sampled via Metropolis-Hastings (with the neighbourhood of γ defined by removing or adding a variable; or by removing *and* adding a variable) is an alternative to the lasso for variable selection: it has different, sometimes better, theoretical properties.

Relevance vector machines explained
T. Fletcher

Relevance vector machines (RVM) consider the bayesian model

$$\begin{aligned} w_j &\sim N(0, \alpha_j^{-1}) \\ \varepsilon_i &\sim N(0, \beta^{-1}) \\ t_i &= w' \phi(x_i) + \varepsilon_i \end{aligned}$$

and estimate its posterior

$$\begin{aligned} w \mid t, \alpha, \beta &\sim N(m, \Sigma) \\ m &= \beta \Sigma \Phi' t \\ \Sigma &= (\text{diag } \alpha + \beta^{-1} \Phi' \Phi)^{-1} \end{aligned}$$

iteratively, by computing m and Σ for given values of α and β , then computing the values of the hyperparameters α, β that maximize the evidence $P(t \mid \alpha, \beta)$:

$$\begin{aligned} \alpha_i &\leftarrow \frac{1 - \alpha_i \Sigma_{ii}}{m_i^2} \\ \beta &\leftarrow \frac{N - \sum_i (1 - \alpha_i \Sigma_{ii})}{\|t - \Phi m\|^2}. \end{aligned}$$

Quite often, $\alpha_i \rightarrow \infty$, *i.e.*, $w_i \rightarrow 0$: the corresponding parameters can be pruned. The remaining parameters are the *relevance vectors*.

Shotgun stochastic search for “large p ” regression
C. Hans et al. (2007)

Shotgun stochastic search (SSS) is a variant of Metropolis-Hastings (MH) to look for a sparse model:

- Let Γ be the set of the best k models examined so far and γ_n the current model;
- In parallel (MCMC is usually sequential), examine all the (or a large number of) neighbours of γ_n ; add them to Γ ; discard the worst models if $|\Gamma| > k$;

- Choose γ_{n+1} as with MH (if you allow for three types of neighbours, from deletion moves γ^- , additions γ^+ and replacement γ^0 , they are unbalanced, especially in high dimensions: $|\gamma^0| \gg |\gamma^+| \gg |\gamma^-|$ – you may want to sample from them separately), but with an acceptance probability that depends on the sum of the scores of the neighbours of the current point and the candidate.

***Statistical model criticism
using kernel two sample tests***
J.R. Lloyd and Z. Ghahramani

The notion of p -value can be generalized to a Bayesian setting:

- Choose a statistic T ;
- Estimate $P[T \leq T_{\text{obs}}]$ using the posterior distribution of T .

It measures how surprising the data still is, even after observing it.

***Random bits regression:
a strong general predictor for big data***
Y. Wang et al.

Penalized regression on random binary features performs well on big data (same idea as echo state networks or extreme learning machines, which are just SVMs with a randomized (instead of universal) kernel).

Random feature maps for dot product kernels
P. Kar and H. Karnick (2012)

SVMs with a high-dimensional embedding space tend to have too many support vectors. One can transform the kernel to reduce the dimension, randomly (as in the Johnson-Lindenstrauss lemma). For instance, if the kernel is of the form $k(x, y) = f(\langle x, y \rangle)$, with $f(x) = \sum_{n \geq 0} a_n x^n$, where the a_n are nonnegative, one can use this expression to build an embedding $Z : \mathbf{R}^d \rightarrow \mathbf{R}^D$, with $\langle Zx, Zy \rangle \approx k(x, y)$:

- Select N randomly with $P[N = n] \propto 1/p^{n+1}$;
- Select $\omega_1, \dots, \omega_n \in \{\pm 1\}^d$ randomly;
- Let $Z_1(x) = \sqrt{a_N p^{N+1}} \prod_j \omega_j' x_j$;
- Do the same for the other coordinates Z_2, \dots, Z_D .

An introduction to random indexing
M. Sahlgren

LSA (latent semantic indexing) builds a word-document matrix and computes its (truncated) SVD to produce a lower-dimensional representation of the words. However, the initial word-document matrix can be too large. Instead, one can use a random projection (Johnson-Lindenstrauss): it can be built incrementally, one document at a time.

***A practical guide
to applying echo state networks***
M. Lukoševičius

Implementation advice for *reservoir computing*.

fastFM: a library for factorization machines
I. Bayer

C library, with Python bindings (contrary to libfm). Factorization machines are not limited to recommendation systems, but can add interactions to any model.

***Tensor machines
for learning target-specific polynomial features***
J. Yang and A. Gittens

Tensor machines are a generalization of factorization machines

$$f(x) = w^0 + \langle w^1, x \rangle + \sum_{p=2}^q \left\langle \sum_{i=1}^r w_1^{p_i} \bullet \dots \bullet w_p^{p_i}, x \bullet \dots \bullet x \right\rangle$$

where $\bullet \dots \bullet$ is the outer product and $w_1^{p_i} \bullet \dots \bullet w_p^{p_i}$ is a rank-1 tensor. The model is fitted by minimizing

$$\frac{1}{n} \sum_i \text{loss}(f(x_i), y_i) + \lambda \|w\|_2^2.$$

(Kar-Karnick features are similar, but w is constrained to be in the subspace spanned by a sufficiently large number of random tensors.)

***Score function features for discriminative
learning: matrix and tensor frameworks***
M. Janzamin et al. (2015)

To build features:

- Estimate the probability distribution $p(x)$ of the (unlabelled) data x ;
- Compute the score functions

$$S_m(x) = (-1)^m \frac{\nabla^m p(x)}{p(x)};$$

- Given labeled data (x, y) , we want some information about $G(x) = E[y|x]$; we can estimate the expectation of its derivatives: $E[\nabla^m G(x)] = E[y S_m(x)]$;
- These are (huge) tensors: compute a low rank approximation (SVD, CP), $E[\nabla^m G(x)] \approx \sum_j u_j^{\otimes m}$ (each tensor product only involves a single vector because the tensors are symmetric);
- Use $\sigma(x \cdot u_j)$ as features where σ is a sigmoid function.

***Picture: a probabilistic programming language
for scene perception***
T.D. Kulkarni et al.

A scene description language can be turned into a probabilistic programming language; do not use a fully-rendered (photorealistic) scene to compute the model with the data pixel by pixel: use features instead (from computer vision or deep learning).

Slice sampling for probabilistic programming
R. Ranca (2015)

To sample from a probability distribution only known up to a multiplicative factor, $p(x) \propto p^*(x)$, **slice sampling** proceeds iteratively:

- Pick $u_t \sim U(0, p^*(x_t))$;
- Pick $x_{t+1} \sim U(\{x : p^*(x) \geq u_t\})$.

StocPy is yet another Python-based probabilistic programming language.

Count-min-log sketch: approximately counting with approximate counters
G. Pitel and G. Fouquier (2015)

Count-min-sketch variant with more precision for low counts, with applications to TD-IDF matrices.

Supervised learning from multiple experts: whom to trust when everyone lies a bit
V.C. Raykar et al. (2009)

Binary forecasts y_{ij} , for many tasks i (e.g., medical diagnostic) for several experts j can be combined (use majority voting as a baseline model) with an EM algorithm, alternatively estimating the forecasts and the reliability of each expert; one can add a Bayesian prior if some experts are known to be more or less reliable.

Invariant backpropagation: how to train a transformation-invariant neural network
S. Demyanov et al.

One expects neural networks to be invariants to some transformations (rotations, translations, etc.). The following weight updates achieve that (you do not have to specify the group action):

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \text{Loss}}{\partial \mathbf{w}} - \beta \frac{\partial}{\partial \mathbf{w}} \left\| \frac{\partial \text{Loss}}{\partial \text{Input}} \right\|^2.$$

Learning classifiers from synthetic data using a multichannel autoencoder
X. Zhang et al.

Use an autoencoder (trained on real data) on surrogate data to make it more realistic.

Deep transform: cocktail party source separation via probabilistic re-synthesis
A.J.R. Simpson

To separate two speech signals (in a mono-aural signal), learn an auto-encoder



[They used a male and a female speaker: their auto-encoder learned to separate gender, rather than speech.]

Explaining and harnessing adversarial examples
I.J. Goodfellow et al. (2015)

For high-dimensional linear models, you can make infinitesimal changes to the input (e.g., ± 1 to each 8-bit pixel of an image) that add up to a large change in the output. To mitigate the problem:

- Use “deep” neural networks (at least one hidden layer);
- Train on a mixture of adversarial and clean examples.

(RBF networks are immune to those adversarial examples, but do not generalize well.)

Neural Turing machine
A. Graves et al.

A differentiable Turing machine can be trained with gradient descent to learn simple algorithms.

Long short-term memory over tree structures
X. Zhu et al. (2015)

LSTM networks can be generalized from sequences to trees.

LSTM: a search space odyssey
K. Greff et al.

In a recurrent net, there is feedback from the output to the input. In an LSTM net, there is feedback from (some of) the inner nodes to (themselves and) the input: they can be trained like RNN (with the same problems).

jvmr: integration of R, Java and Scala
D.B. Dahl et al.

Embed R in Scala or Java, or the reverse.

Bayesian estimation supersedes the t test
J.K. Kruschke (2013)

For a Bayesian T test, check the BEST R package.

Cubist models for regression
M. Kuhn et al. (2012)

Cubist models generalize decision trees:

- There is a model in each node, and each leaf (or node) is shrunk towards its parent;
- One can add boosting and/or shrink each forecast towards its nearest neighbours (in the training set or the sample to forecast).

1. Several presentations applied social network analysis (SNA) methods to financial networks.

PageRank on a graph weighted by cross-correlations can identify leaders. [Some use the HITS algorithm instead, to identify leaders and followers, extracting information from leaders and investing in followers.]

If the nodes i of a graph represent financial entities, each with a risk C_i , one can compute the following graph metrics:

$$\text{Fragility} = \frac{E[\text{degree}^2]}{E[\text{degree}]}$$

$$\text{Risk score} = \sqrt{C'AC}$$

where A is the incidence matrix The risk score can be decomposed with Euler's formula:

$$\text{Risk score} = \sum \frac{\partial \text{Score}}{\partial C_i} C_i.$$

One can then compute

$$\text{Criticality} = C \odot x$$

where x is the eigenvector centrality (not unlike PageRank) and \odot the elementwise product, and

$$\text{Spillover}_{ij} = \frac{\partial}{\partial C_i} \frac{\partial \text{Score}}{\partial C_j}.$$

A graph (e.g., interbank lending market, in some country) can be decomposed into *core and periphery*, and compute various graph metrics (density, betweenness centrality, closeness centrality, transitivity, average path length, eigenvalues, etc.) over time.

One can apply network analysis (plots, centrality) on raw (SWIFT) financial transactions.

One can compute graph metrics (centrality, etc.) on graphs learnt with **bnlearn**.

2. Higher moments (up to the fifth...) were mentioned a couple of times.

Use differential evolution (**DEoptim**) to optimize the expected utility of a portfolio (using the sample distribution of past returns). The derivatives of the utility function should alternate in sign. They all have names:

$U' > 0$	Non-satiation (high returns)
$U'' < 0$	Risk aversion (low risk)
$U''' > 0$	Prudence (positive skew)
$U'''' < 0$	Temperance (low kurtosis)
$U''''' > 0$	Edginess.

Use, for instance, $U(\text{wealth}) = \text{wealth}/(c + \text{wealth})$.

Investors have a preference for high odd moments and low even moments. The **HighFreq** package estimates them from OHLC data (not unlike all the volatility estimators in **TTR**).

3. A *partially auto-regressive* process is the sum of a stationary AR process and a random walk; it can be estimated with a Kalman filter. Caveats: the tests have low power; the AR component may just capture the market microstructure (e.g., the bid-ask bounce). This is implemented in the **partialAR** package (use the cointegration residuals, from **egcm**).

To identify change points in a time series, find the partition of the indices that maximizes the sample energy distance

$$\mathcal{E}(X, Y) = 2E|X - Y|^\alpha - E|X_1 - X_2|^\alpha - E|Y_1 - Y_2|^\alpha$$

$$\hat{\mathcal{E}}(X, Y) = \frac{2}{\#B} \sum_B |x_i - x_j|^\alpha +$$

$$- \frac{1}{\#W_x} \sum_{W_x} |x_i - x_j|^\alpha - \frac{1}{\#W_y} \sum_{W_y} |y_i - y_j|^\alpha$$

The **e-cp3o** algorithm speeds up the computations by removing points that are unlikely to be change points and reducing the size of B , W_x , W_y . This is implemented in **ecp::ecp3o**.

For large but sparse VARX models (VAR with exogenous variables), add a (nested group) lasso penalty. The Hierarchical VAR (HVAR) model also adds lasso penalties to the lags. This is implemented in the **BigVAR** package.

By combining bits of models that are not often used together, one can build rather complicated models, e.g., a regime switching cointegration model with time-varying transition probabilities

To detect multivariate outliers, use a robust covariance matrix, e.g., the MCD (minimum covariance determinant). Iteratively-reweighted MCD gives the correct false positive rate. This is implemented in the **CeriolliOutlierDetection** package.

4. The **PortfolioAnalytics** package provides many portfolio optimization approaches, including Meucci's fully flexible views or Almgren and Chriss's portfolios from sorts.

Flexible asset allocation with stepwise correlation rank suggests to build a portfolio one asset at a time, adding the asset with the best

$$w_1 \times \text{momentum rank} + w_2 \times \text{volatility rank} +$$

$$w_3 \times \text{correlation rank},$$

where the correlation is the average correlation with the assets already in the portfolio.

It is safer to build a portfolio from a few (smart beta) ETFs than from a large number of stocks: lower estimation error and fewer parameters lead to better out-of-sample performance.

Taxes have an impact on the optimal multi-period 2-asset portfolio (Kelly principle).

5. mVaR and mES (Cornish-Fisher value at risk and expected shortfall) are estimators: we can compute their variance and bias (a computer algebra system

(CAS) is useful – they used the Matlab symbolic toolbox, but you may want to check SAGE, Maxima or Yacas) and compare them with a maximum likelihood estimator (asymptotically no bias and lowest variance). Also check `qrntools::ARA` and the `SharpeR` package. `mVaR` and `mES` are bad for: low α , fat tails, large samples.

One can still decompose the VaR into a sum of contributions (risk factors, managers) in the presence of non-linear assets (e.g., with the delta-gamma approximation).

One can compute an upper bound on the expected Sharpe ratio; it is unclear if/how it depends on the number of assets.

6. The `highfrequency` package provides functions to align non-synchronous time series (wait until there has been a new observation for all time series), compute volatility or jump estimators (realized bipower variation, MedRV, ROWVar), tests for the presence of jumps, etc.

Here is a stylized fact for high frequency data:

Number of transactions \propto USD Volume $^{2/3} \times$ Volatility.

The `creditr` package provides CDS computations, with a Shiny interface, not unlike the Bloomberg or Markit CDS screen/calculator.

7. The `Rborist` package is similar to `RandomForest`, but faster; it can use several cores (or a GPU).

The `irlba` package computes the fast truncated SVD.

8. `data.table` still has an edge on `dplyr`:

- The data is indexed;
- It provides fast aggregation, and ordered joins (aka rolling joins, `lof`);
- `fread` is a fast CSV reader (but `h2o.importFile` is even faster).

The development version of DBI (finally) allows for parametric queries (prepared statements). The `readr` package provides faster functions to read CSV files (but `fread` is even faster). The `httr`, `xml2`, `rvest`, `jsonlite` packages were also mentioned; `dplyr`, `reshape2`, etc. were not.

`Rcpp` provides better integration with RStudio, annotations (e.g. `// [[Rcpp::export]]`), C++11 (closures, type inference), direct access to Boost even on Windows (BH).

`Rblpapi` is a newer interface to Bloomberg, in C++ rather than Java. It does not work well on Windows.

Some suggest to describe the data pipeline in a purely declarative way; to facilitate deployment, this should also include the data cleaning part.

The `simsalapar` package can help you (easily) parallelize computations – when you have to run the same computations/simulations many times, with different parameter values. An example for VaR computations with nested archimedean copulas

(`copula::onacopula`) was given. [I prefer to use `doParallel` (and `foreach`) directly.]

The computation of the expected improvement in dominated hypervolume of Pareto front approximations
M. Emmerich et al. (2008)

For multi-objective Bayesian optimization, replace the expected improvement (EI) with the expected increase in the hypervolume dominated by the Pareto set. While a Monte Carlo approach is possible, it can be computed exactly: decompose the space into boxes, each corresponding to a different shape for the new efficient frontier, and compute the expected improvement conditional on the new point being in a given box.

Faster computation of expected hypervolume improvement
I. Hupkens et al. (2014)

Expensive multiobjective optimization for robotics
M. Tesch et al. (2013)

Bayesian multiobjective optimization to steer snake robots, to optimize both speed and head camera stability: replace the expected improvement (EI) with the expected improvement in (dominated) hypervolume (EIHV). Also gives another test problem, built from the Branin function.

A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning
E. Brochu et al. (2010)

Good review article.

ParEGO: a hybrid algorithm with online landscape approximation for expensive multiobjective optimization
J. Knowles (2004)

The following multiobjective optimization algorithm is more economical (in function evaluations) than NSGA-II:

- Initialize the population with a Latin hypercube;
- Take a weight vector $\lambda \in \Delta_n$ at random (a new one for each iteration, possibly with some of the coordinates set to zero);
- Consider the objective function

$$f(x) = \text{Max}_i \lambda_i f_i(x) + \rho \sum_i \lambda_i f_i(x);$$

- Model it using a Gaussian process (if there are more than 80 points in the population, take 80 at random);
- Use some optimization algorithm (say, Nelder-Mead) to maximize the expected improvement;

- Add the new point to the population;
- Iterate.

The paper also provides a few test functions.

***Multiobjective optimization
on a budget of 250 evaluations***
J. Knowles and E.J. Hughes (2005)

ParEGO performs better than a binary-search-based algorithm:

- Pick a point at random in the largest hypercube, and split the hypercube at that point in the direction that yields the most “cube-like” sub-spaces;
- Idem with a point close to a “good point”, for some random aggregation of the scores, e.g. $\sum_i \lambda_i f_i(x)$, $\lambda \sim U(\Delta_n)$;
- Alternate between those steps with a predefined exploration/exploitation ratio.

***Multiobjective optimization
of urban wastewater systems using ParEGO:
a comparison with NSGA-II***
G. Fu et al. (2008)

Another comparison of ParEGO and NSGA-II.

***Efficient global optimization
of expensive black-box functions***
D.R. Jones et al. (1998)

Not very different from DACE: start with a Latin hypercube; check the goodness of fit of the model (and transform the variables if needed); compute the expected improvement in closed form.

***Efficient global optimization (EGO)
for multi-objective problem and data mining***
S. Jeong and S. Obayashi (2005)

Another EGO example.

BOA: the Bayesian optimization algorithm
M. Pelikan et al. (1999)

Variant of CMA-ES with Bayesian networks (suitable for discrete variables) instead of a Gaussian distribution.

***Bayesian optimization algorithms
for multi-objective optimization***
M. Laumanns and J. Ocenasek (2002)

The multiobjective Bayesian optimization algorithm works as follows:

- Keep an archive of best solutions (all those that ε -dominate the solutions seen so far, but allow some dominated solutions to have at least k of them);
- Model the archive, not unlike CMA-ES, but with decision trees (one for each variable – Bayesian networks proved too complex) instead of a Gaussian distribution;
- Sample candidates from this model.

***Multi-objective
Bayesian optimization algorithm***
N. Khan et al. (2002)

BOA can be generalized to multiobjective optimization: use non-dominated sorting and the crowding distance to select the n best solutions.

***Multiobjective evolutionary algorithm
test suites***
D.A. van Veldhuizen and G.B. Lamont (1999)

***Scalable test problems for evolutionary
multi-objective optimization***
K. Deb et al. (2001)

***Combining multiobjective optimization
and Bayesian model averaging to calibrate
forecast ensembles of soil hydraulic models***
T. Wöhling and J.A. Vrugt (2008)

Bayesian optimization with Gaussian processes replaced by a (BMA) ensemble of domain-specific (PDE) models.

***Bayesian optimization
with inequality constraints***
J.R. Gardner et al. (2014)

Constrained Bayesian optimization,

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & f(x) \\ \text{Such that} & g(x) \geq 0, \end{array}$$

where both the objective f and the constraints g are expensive to compute, is a straightforward generalization of Bayesian optimization: replace the expected improvement $E[(f(x^+) - \hat{f}(x))_+]$ with the expected constrained improvement $E[\mathbf{1}_{\hat{g}(x)} \cdot (f(x^+) - \hat{f}(x))_+]$. One often assumes conditional independence: $\hat{g}(x) \perp\!\!\!\perp \hat{f}(x) \mid x$.

***Pareto front modeling for sensitivity analysis
in multi-objective Bayesian optimization***
R. Calandra et al. (2014)

In multiobjective Bayesian optimization (MOBO), do not return a discrete set of solutions but, since a model was used for the optimization, a model of the Pareto front.

Design and analysis of computer experiments
J. Sacks et al. (1989)

One of the first papers on Bayesian optimization.

*A fast and elitist
multiobjective genetic algorithm: NSGA-II*
K. Deb et al. (2002)

The non-dominated sorting genetic algorithm II (NSGA-II) solves multi-objective optimization problems, *i.e.*, estimates their non-dominated front.

1. The candidate solutions are sorted by first finding the non-dominated front, removing it, and iterating. The **non-dominated rank** can be computed efficiently:
 - For each solution, compute the domination count, *i.e.*, the number of solutions that dominate it;
 - For each solution, compute the set of solutions it dominates;
 - The first non-domination front solutions have domination count zero: remove it;
 - Update the domination counts and the domination sets;
 - Iterate.
2. To preserve diversity, look at the **crowding distance** of each solution, defined as the average side length of the cuboid around it with (half) its vertices among the other solutions.
3. Define a partial order on the solutions:
 - Prefer solutions with a lower non-domination rank;
 - If the ranks are equal, prefer solutions with a higher crowding distance.
4. One can add constraints:
 - Prefer feasible solutions;
 - When two solutions are infeasible, prefer that with the smaller constraint violation.

The paper also contains reference problems.

*GPfit: an R package
for Gaussian process model fitting
using a new optimization algorithm*
B. MacDonald et al. (2013)

When fitting a Gaussian process on non-noisy data, with a Gaussian correlation function

$$R_{ij} = \text{Cor}(y_i, y_j) = \prod_k \exp -\theta_k |x_{ik} - x_{jk}|^2,$$

$\theta_k > 0$, the correlation matrix can be ill-conditioned (some of the points are too close), and the likelihood has local extrema very close to zero. One can replace R with $R + \delta I$, with the smallest nugget δ that makes the matrix well-conditioned [this is the idea behind ridge regression; this is also what you naturally do when the eigenvalues are too close to zero], and reparametrize θ as $\theta = \exp \phi$.

Check the R packages **GPfit** (no noise), **tgp** (noise) or **mlegp** (noise, numerically unstable).

Also mentions **lhs::maximinLHS** (random but well-dispersed points).

*News and monetary shocks
at a high frequency: a simple approach*
T. Matheson and E. Stavrev (2014)

Given a bivariate time series X , one can estimate a VAR(1) model $X_t = \alpha + \beta X_{t-1} + \varepsilon_t$ using OLS.

To interpret the residual (*i.e.*, to estimate a structural VAR (SVAR) model), one can try to write it as an MA(1) process, $\varepsilon_t = A\eta_t$ where $\eta \sim N(0, 1)$. The innovations η represent independent (Gaussian, standard) shocks.

There is not enough information to estimate both A and η : if $\Sigma = \text{Var} \varepsilon$ has Choleski decomposition $\Sigma = PP'$, then $A = P$ is a solution, but so is $A = PU$, for $U \in O_2$. One typically adds linear constraints on A to ensure that there is only one solution, *e.g.*, A lower-triangular.

Instead, one can impose constraints on the signs of (some) of the coefficients) of A , *e.g.*,

$$\begin{pmatrix} + & + \\ + & - \end{pmatrix}.$$

This is still insufficient to determine A , but it is sufficient for a Bayesian approach: one can compute the posterior distribution of A (using, *e.g.*, a uniform prior on O_2). The authors use rejection sampling [but since the elements of O_2 admit a simple parametrization, exact computation should be possible – rejection sampling poses even larger problems in higher dimensions]. Since $O_2 \approx \mathbf{S}^1 \amalg \mathbf{S}^1$, it is not straightforward to extract an estimator from the posterior distribution.

*How do fiscal and technology shocks
affect real exchange rates?
New evidence for the United States*
Z. Enders et al. (2008)

More details on SVAR models with sign restrictions: allowance for small deviations to the sign constraints, use of the posterior distribution to test the sign of the non-constrained elements.

*Scaling log-linear analysis
to datasets with thousands of variables*
F. Petitjean and G.I. Webb (2015)

Given M qualitative variables V_1, \dots, V_M , **log-linear analysis** (LLA) greedily builds a model

$$P[V_1 = a_1, \dots, V_M = a_M] = u + \sum_i u_i(a_i) + \sum_{i < j} u_{ij}(a_i, a_j) + \dots$$

only adding a term if it leads to a significant difference.

At each step, it is not necessary to re-examine all the (tuples) of variables: one can identify many of those

that remain insignificant and only re-evaluate the others.

Implementation: Chordalysis (Java, GPL).

A statistically efficient and scalable method for log-linear analysis of high-dimensional data
F. Petitjean et al. (2014)

Use minimum message length (MML) instead of goodness-of-fit χ^2 tests.

Scaling log-linear analysis to high-dimensional data
F. Petitjean et al. (2013)

Considering only decomposable models (if an interaction between k variables is present, then so are all of its subsets, and the corresponding graph is *chordal*) leads to more efficient χ^2 computations.

Unsupervised learning of acoustic features via deep canonical correlation analysis
W. Wang et al.

Canonical correlation analysis (CCA) looks for highly correlated linear combinations between two groups of variables, X and Y :

$$\begin{array}{ll} \text{Find} & u, v \\ \text{To maximize} & \text{Cor}(u'X, v'Y). \end{array}$$

More generally (if X and Y are centered),

$$\begin{array}{ll} \text{Find} & U, V \\ \text{To maximize} & \text{trace } U'XY'V \\ \text{Such that} & U'XX'U = nI \\ & V'YY'V = nI. \end{array}$$

This is equivalent to minimizing

$$\frac{1}{n} \|U'X - V'Y\|_F^2 + \lambda \|U\|_F^2 + \lambda \|V\|_F^2.$$

One can replace X and Y with their embedding in a reproducing kernel Hilbert space (RKHS) or a deep neural net.

Tensor factorization via matrix factorization
V. Kuleshov et al. (2015)

To compute an approximate CP decomposition of a symmetric 3-tensor, evaluate it on a set of random vectors to get symmetric matrices $(M_\ell)_\ell$ and jointly diagonalize them, *i.e.*, find U orthogonal such that $UM_\ell U'$ be almost diagonal (use an L^2 penalty on the off-diagonal elements – it is possible to build U iteratively, as a product of simple (rotation) matrices). This can be generalized to higher-order tensors or to non-symmetric tensors – consider

$$\begin{pmatrix} 0 & M' \\ M & 0 \end{pmatrix}.$$

Fastfood – approximating kernel expansions in loglinear time
Q. Le et al. (2013)

Forecasts from kernel methods require the evaluation of the kernel on a large number of points (e.g., for SVM, one needs

$$f(x) = \sum_{y \text{ support vector}} \alpha_y k(y, x),$$

and the number of support vectors grows linearly with the data).

A kernel can be written

$$k(x, y) = \sum \lambda_i \phi_i(x) \phi_i(y).$$

It can be approximated by drawing i_1, \dots, i_n with $p(i) \propto \lambda_i$ and considering $k(x, y) \approx \sum_{k=1}^n \phi_{i_k}(x) \phi_{i_k}(y)$.

For Gaussian radial basis functions (RBF), one can use $\phi_j(x) = \exp(i(Zx)_j)$ as eigenfunctions, where Z is a Gaussian random matrix.

This matrix is large, but may be replaced with $SHG\Pi HB$ where B has random ± 1 on the diagonal, G has random Gaussian numbers on the diagonal, S is a (diagonal) scaling factor, Π is a random permutation matrix and H is a Walsh-Hadamard matrix:

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_{2d} = \begin{pmatrix} H_d & H_d \\ H_d & -H_d \end{pmatrix}.$$

Speeding up convolutional neural network using fine-tuned CP-decomposition
V. Lebedev et al. (2015)

Convolutional neural networks (CNN) can be sped up by replacing the 4D convolution kernel with a low-rank approximation.

An entropy-based early-warning indicator for systemic risk
M. Billio et al. (2015)

Use the entropy

$$-\sum p_i \log p_i \quad (\text{Shannon})$$

$$\frac{1}{\alpha - 1} \left(1 - \sum p_i^\alpha \right) \quad (\text{Tsallis})$$

$$\frac{1}{1 - \alpha} \log \sum p_i^\alpha \quad (\text{Rényi})$$

of

- The marginal expected shortfall

$$\text{MES}_i = E[r_i \mid r_{\text{market}} < \text{VaR}_{\text{market}}^{5\%}];$$

- ΔCoVaR , where

$$\Delta\text{CoVaR}_{m,i}^\alpha = \text{CoVaR}_{m,i}^\alpha - \text{CoVaR}_{m,i}^{1/2}$$

$$\text{CoVaR}_{m,i}^\alpha = \text{VaR}^\alpha[r_{\text{market}} \mid r_i = \text{VaR}_i^\alpha];$$

- The number of edges in the Granger causality graph of asset returns

to predict crises (in the Finance industry).

New ranks for even-order tensors and their application in low-rank tensor optimization
B. Jiang et al. (2015)

For tensors, the notion of rank is complicated.

The Black-Litterman approach: original model and extensions
A. Meucci (2010)

The Black-Litterman model

π	equilibrium market returns
$\mu \sim N(\pi, \tau\Sigma)$	<i>expected</i> market returns
$X \sim N(\mu, \Sigma)$	market returns
$P'\mu \sim N(v, \Omega)$	view on the expected returns

is needlessly complicated and even counter-intuitive (the limiting cases $\Omega \rightarrow 0$ or $\Omega \rightarrow \infty$ are not those the end-user will expect). Instead, one can specify views on the market returns:

$$X \sim N(\mu, \Sigma)$$

$$P'X \sim N(v, \Omega).$$

To compute the posterior distribution, reformulate the model as

$X \sim N(\mu, \Sigma)$	returns
$Z \sim N(0, \Omega)$	error on the views
$v = P'X + Z$	views,

compute the joint distribution of $(X, Z, P'X + Z)$ and then the conditional distribution $X \mid P'X + Z = v$.

A fuzzy R code similarity search detection algorithm
A. Bartoszek and M. Gagilewski

To measure the similarity (plagiarism, copy-paste) between two functions:

- Compare the edit distance μ_1 (`utils::adist`) between the strings (normalized with `deparse(paste(text=f))`);
- Consider the blocks of code as bags of functions (indeed, `<-`, `:`, `+`, `for` are functions) and compute some variant of Jaccard similarity, e.g.,

$$\mu_2(x, y) = \frac{\sum \text{Min}(x_i, y_i)}{\sum x_i + y_i}$$

where \mathbf{x} and \mathbf{y} are the count vectors;

- Compute the abstract syntax tree (AST) (`utils::getParseData`), ignore names and values (`SYMBOL`, `NUM_CONST`) and compute the total length μ_3 of the common substrings above some minimum length;
- Combine those three measures using a training set and logistic regression.

Other plagiarism detection tools include: MOSS, JPlag, GPlag.

Detecting similarity of R functions via a fusion of multiple heuristic methods
A. Bartoszek and M. Gagilewski

To measure code similarity, use asymmetric measures, similar to the previous three ones, but normalized by dividing by the length of the first function, e.g.,

$$\mu_2(x, y) = \frac{\sum \text{Min}(x_i, y_i)}{\sum x_i}.$$

Add another measure,

$$\mu_4 = \frac{\#V(H_{ij})}{\#V(G_i)}$$

where G_i is the *program dependence graph* (PDG: the vertices are the vertices of the AST; there are two types of edges, corresponding to the control flow and to data dependencies) of f_i , and H_{ij} is the largest common (isomorphic) subgraph of G_i and G_j (computed with the McGregor heuristic algorithm, from the Boost C++ library).

An industrial-strength audio search algorithm
A.L.C. Wang

To fingerprint audio files:

- Compute the time×frequency spectrogram;
- Identify peaks; ensure they are approximately uniformly distributed;
- Use rectangular regions of this constellation of peaks as fingerprint.

Search is then an alignment problem.

A comparison of fractal trees to log-structured merge (LSM) trees
B.C. Kuszmaul (2014)

A *fractal tree index* is a B-tree with a buffer at each node: only when the buffer is full is it sent to lower nodes. This requires more memory, but greatly reduces disk writes. [Nothing seems to justify the work “fractal”: “lazy” would have been a better name.]

LSM trees store runs of data, *i.e.*, lists of key-value pairs sorted in key order (some big data databases keep recently-inserted data in memory, sorted, as long as it fits, and then write it to disk, each time in a different file). These runs can be arranged in a tree.

Data structures for text sequences
C. Crowley (1998)

To store and manipulate text, with insert and delete operations (as in a text editor), check the following data structures:

- Array;
- Array with a gap in the middle (corresponding to the current position);
- Doubly-linked list of characters, or lines, or fixed-length (partially-empty) buffers;

- PieceTable: two buffers, one read-only (the original file), one append-only, with an array of pointers indicating which span of which buffer to use in which order.

An efficient natural neighbour interpolation algorithm for geoscientific modelling

H. Ledoux and C. Gold

Move in the space of triangulations using flipping movements $\diamond \mapsto \triangleleft$ until you reach the Delaunay triangulation.

A stable and fast implementation of natural neighbor interpolation

L. Liang and D. Hale (2010)

Java implementation: `edu.mines.booles.jtk.interp.`

Differential privacy and machine learning: a survey and review

Z. Ji et al. (2014)

Differential privacy adds noise to ensure that changes to a single individual have a negligible impact. For instance, one can add Laplace noise to the result $f(D)$, with the amplitude of the noise proportional to

$$\text{Max}_{d(D_1, D_2)=1} \|f(D_1) - f(D_2)\| \text{ or } \text{Max}_{d(D, D')=1} \|f(D) - f(D')\|$$

(in the second case, only D' varies, but you need some smoothing).

Similarly, to compute $\text{Argmax}_a f(D, a)$ (e.g., to maximize a likelihood), one can sample from a Laplace distribution

$$p(a) \propto \exp \frac{\varepsilon f(D, a)}{2 \text{Max}_{\text{dist}=1} \|\Delta f\|}$$

For most machine learning algorithms, adding noise to the data and/or the loss function and/or the result ensures differential privacy.

DREAM_(D): an adaptive Markov chain Monte Carlo simulation algorithm to solve discrete, non-continuous, and combinatorial posterior parameter estimation problems

J.A. Vrugt and C.J.F. Ter Braak (2011)

Differential evolution can be used as a proposal distribution in an adaptive Metropolis simulation with several MCMC chains (DREAM).

For discrete states, one could try to round the results, but this only works with ordered discrete variables. Instead, one can take two chains at random, two coordinates, and swap them.

Discrete MCMC can also be used for optimization or satisfiability problems (e.g., Sudoku). [How different is it from simulated annealing and genetic algorithms?]

A directional multivariate value at risk

R. Torres et al.

The value-at-risk (VaR) can be generalized to a multivariate setup by considering the set $Q_\alpha = \{\mathbf{x} : P[X \leq x] = \alpha\}$ or, if we prefer a single vector, the intersection $Q_\alpha \cap E[X] + \text{Span } \mathbf{u}$, where $\mathbf{u} = (1, \dots, 1)'$ is the vector in the middle of the positive quadrant. This can be generalized to an arbitrary quadrant. [The authors mistakenly claim that the quadrant is entirely determined by \mathbf{u} .]

Shortfall deviation risk: an alternative for risk measurement

M.B. Righi and P.S. Ceretta

There are many, many variants of value at risk (VaR) and expected shortfall (ES). Here is yet another one:

$$\text{ES} = E[\text{loss} \mid \text{loss} \geq \text{VaR}]$$

$$\text{SDR} = \text{ES} + \sigma[\text{loss} \mid \text{loss} \geq \text{ES}]$$

A robust statistics approach to minimum variance portfolio optimization

L. Yang et al.

Estimate the variance matrix as the fixed point $V(\rho)$ of

$$V(\rho) = (1 - \rho) \frac{1}{n} \sum_i \frac{x_i x_i'}{\frac{1}{n} x_i' V(\rho)^{-1} x_i} + \rho I_N$$

and choose ρ to minimize the (ex post) variance portfolio (Tyler's robust M estimator).

Tornadoes and related damage costs statistical modeling with a semi-Markov approach

C. Corini et al. (2015)

Tornadoes can be modeled using a *Markov renewal process* (J, T)

J_n = (discrete) intensity of the n th tornado
(Fujita scale)

T_n = (discrete) time of the n th tornado.

One can also consider

Z_t = intensity of the last tornado

B_t = time since the last tornado.

Both (J, T) and (Z, B) are Markov; Z is a *semi-Markov* process.

Hawkes processes in finance

E. Bacry et al.

Survey: definitions, properties, applications, estimation, simulation (either one event at a time, as for inhomogeneous Poisson processes, or by tracking the parenthood tree: first, generate events coming from the background rate, then their children, then their grand-children, etc.).

Dirac processes and default risk
C. Kenyon and A. Green (2015)

Since the short rate r is always used in an integral, $\int_{t_1}^{t_2} r(s)ds$, it does not have to be a stochastic process: it can be a generalized process. A *Dirac process* is a process of the form $\sum_i \delta_{T_i}$, where the T_i are the arrival times of a Poisson process and δ_a is the Dirac mass in a . They can be used in option pricing.

The Horseshoe+ estimator of ultra-sparse signals
A. Bhadra et al.

Sparse signals can be recovered by thresholding, using the hierarchical model

$$\begin{aligned}\theta_i &\sim (1 - \mu)\delta_0 + \mu N(0, \psi^2) \\ X_i &\sim N(\theta_i, 1).\end{aligned}$$

The Horseshoe prior is another hierarchical model:

$$\begin{aligned}\lambda_i &\sim \text{Half-Cauchy}(\tau) \\ \theta_i &\sim N(0, \lambda_i^2) \\ X_i &\sim N(\theta_i, 1).\end{aligned}$$

The Horseshoe+ prior is yet another one:

$$\begin{aligned}\eta_i &\sim \text{Half-Cauchy}(1) \\ \lambda_i &\sim \text{Half-Cauchy}(\eta_i \tau) \\ \theta_i &\sim N(0, \lambda_i^2) \\ X_i &\sim N(\theta_i, 1).\end{aligned}$$

Hierarchical Bayesian survival analysis and projective covariate selection in cardiovascular event risk prediction
T. Peltola et al.

Prefer the Horseshoe prior to the Laplace one (lasso): it does not shrink large coefficients too much. The resulting (Bayesian, posterior) model is not sparse, but one can find a sparse model with similar forecasts.

Moments determine the tail of the distribution (but not much else)
B.G. Lindsay and P. Basak (2000)

$$\sup_{\substack{F \text{ and } G \text{ have} \\ \text{the same first} \\ p \text{ moments}}} |F(x) - G(x)| \leq w_p(x)$$

$$w_p(x) = (1 \quad x \quad \cdots \quad x^p) \begin{pmatrix} 1 & m_1 & \cdots & m_p \\ m_1 & & & \vdots \\ \vdots & & & \vdots \\ m_p & \cdots & \cdots & m_{2p} \end{pmatrix} \begin{pmatrix} 1 \\ x \\ \vdots \\ x^p \end{pmatrix}$$

Distinguishing cause from effect using observational data: methods and benchmarks
J.M. Mooij et al.

Comparison of bivariate causal discovery algorithms, additive noise models (compare the complexity of the linear models $y \sim x$ and $x \sim y$) and information-theoretic causal inference (compare the flatness or entropy of the margins – for non-linear relations, the flatter is the cause) on a benchmark dataset.

A fast unified algorithm for solving group-lasso penalized learning problems
Y. Yang and H. Zou (2014)

A faster algorithm for the *group lasso*

$$\hat{\beta} = \underset{\beta}{\text{Argmin}} \text{Loss}(\beta) + \sum_{g \in \text{Groups}} \|\beta_g\|_2$$

(this is $\|\cdot\|_2$, not $\|\cdot\|_2^2$), when the loss function has a bounded second derivative (or, more generally, a Lipschitz first derivative: least squares, squared hinge loss, huberized hinge loss).

Implemented in R in `gglasso`. Also check `grplasso`.

Archimedian-based Marshall-Olkin distributions and related copula functions
S. Mulinacci

The generalized Marshall-Olkin copula is the copula of $(\text{Min}(X_1, X_3), \text{Min}(X_2, X_3))$, where (X_1, X_2, X_3) has an archimedian copula and (for instance) exponential margins. The random variables X_1 and X_2 are the failure times of two machines, X_3 is a systemic failure time (it entails the failure of both machines), the archimedian copula represents a common factor that influences all three types of failures in the same way (e.g., the maintenance of the power plant).

Fully automated variational inference of differentiable probability models
A. Kucukelbir et al.

Variational Bayes in Stan: let X denote the observed variable, Z the hidden variable; $p(X, Z)$ is known, but $p(Z|X)$ is too complicated. One can look for $q(Z)$, Gaussian, close to $p(Z|X)$ for the Kullback-Leibler distance, estimated by gradient descent.

blowtorch: an R package for on-line constrained optimization
L. Petito and S. Pollack

The `blowtorch` package solves the equality-constrained optimization problem

$$\begin{aligned}\text{Find} & \quad x \\ \text{To maximize} & \quad f(x) \\ \text{Such that} & \quad G(x) = 0\end{aligned}$$

using Lagrange multipliers

$$\Lambda(x, \lambda) = f(x) - \lambda \cdot G(x)$$

and stochastic gradient descent (SGD) to minimize $\|\nabla\Lambda\|_2^2$. The user must provide f', f'', G, G', G'' ; it can be used as a pre-processing step before CG (conjugate gradient) or BFGS.

***Stochastic dual coordinate ascent methods
for regularized loss minimization***
S. Shalev-Shwartz and T. Zhang

One can replace gradient descent

$$\begin{array}{ll} \text{Find} & w \\ \text{To minimize} & \frac{1}{n} \sum_i \phi_i(w'x_i) + \frac{\lambda}{2} \|w\|^2 \end{array}$$

with dual coordinate ascent

$$\begin{array}{ll} \text{Find} & x \\ \text{To maximize} & \frac{1}{n} \sum -\phi_i^*(-\alpha_i) - \frac{\lambda}{2} \left\| \frac{1}{\lambda n} \sum \alpha_i x_i \right\|^2 \end{array}$$

For faster convergence, start with SGD, then use stochastic DCA.

***The role of Occam's razor
in knowledge discovery***
P. Domingos

There are two versions of Occam's razor:

1. If the generalization error is the same, prefer simpler models: they are easier to understand and explain;
2. If the training error is the same, prefer the simpler model: the generalization error will be lower.

The second is wrong:

- In a regression, one should not round the coefficients, even though “ $x + 2$ ” is simpler than “ $1.17x + 1.98$ ”;
- The VC dimension is a better indicator of the generalization error than the number of parameters;
- The number of models among which the final model is chosen is more important than its complexity (multiple testing);
- Insanely complex models work well: penalized regression, the kernel trick which inflates the dimension of the feature space, ensemble models, deep learning.

***Estimating the algorithmic complexity
of stock markets***
O. Brandouy et al. (2015)

The Kolmogorov complexity $K(s)$ of a binary string s is the minimum length of a program (in some fixed language) that outputs s . An (infinite) string s is *random* (this is *not* a probabilistic notion) if

$$\exists c \quad \forall n \quad K(s_{1:n}) \geq n - c.$$

One can estimate the randomness of a (long) string by looking at its compressibility, with algorithms such as Huffman, RLE or Paq808. To apply this idea to time series, iteratively remove the visible structure, try to compress, and iterate as long as the signal remains compressible:

- Remove the visible structure from the prices: compute the log-returns;
- Remove the visible structure (the distribution of returns): uniformize the returns;
- Remove the visible structure (volatility clustering): uniformize the returns on a moving window;
- Discretize, compress.

What remains is not clear: it could be market microstructure (e.g., tick size) or the artefacts of the structure-removing procedures.

Liquidity crises on different time scales
F. Corradi et al. (2015)

The notion of liquidity depends on the time scale: the depth and breadth of the limit order book around 30 seconds; the speed at which the order imbalance disappears, around 15 minutes.

***Iterated prisoner's dilemma contains strategies
that dominate any evolutionary opponent***
W.H. Press and F.J. Dyson (2012)

In the iterated prisoner's dilemma (IPD), a longer (but still finite) memory does not give any advantage (over a 1-period memory). Strategies can be modeled by a Markov chain. A player can decide on his/her opponent's rewards (but not on his/her own).

***Optimal thresholding of classifiers
to maximize F1 measure***
Z.C. Lipton et al.

Extreme learning machines
E. Cambria and G.B. Huang (2013)

ELMs (or echo networks) are neural nets whose hidden layers are initialized at random; they can be seen as random kernel embeddings.

Text understanding from scratch
X. Zhang and Y. LeCun

Convolutional neural networks can be used on streams of characters (not words) for text classification (6 convolutional layers, 3 fully connected ones, 2 dropout units); the characters are encoded as m -dimensional boolean vectors.

The training data can be augmented with synonyms (mytheas, from LibreOffice, itself from Wordnet). The approach also works with non-alphabetic languages using romanization (for Chinese: pinyin and jieba)

It works better than a bag-of-words (logistic regression on the 5000 most frequent words) or a bag of centroids (idem, after applying k -means, $k = 5000$, to word2vec).

***Delving deep into rectifiers:
surpassing human-level performance
on ImageNet classification***
K. He et al.

Rectified linear units (ReLU) $f(y) = y_+$ can be generalized to parametric rectified linear units $f(y) = y_+ - ay_-$, where a is learned (and not penalized – like biases). The weights can be initialized randomly.

Online passive-aggressive algorithms
K. Crammer et al. (2006)

SVMs or regressions can be estimated online: if the new observation is correctly classified (with high enough a margin), do not update the model (passive step), otherwise, update it a little (aggressive step).

The relaxed online maximum margin algorithm
Y. Li and P.M. Long (2001)

SVMs can be fit online (*i.e.*, one observations at a time).

***Why does deep learning work?
A perspective from group theory***
A. Paul and S. Venkatasubramanian (2015)

In autoencoders formed a group

$$\begin{aligned} G &= \{\text{autoencoder}\} \\ X &= \{\text{possible inputs}\} \\ Y &= \{\text{training data}\} \subset X, \end{aligned}$$

with G acting on X , learning would be a random walk looking for $g \in \text{Stab } Y$. This can be made rigorous.

***Evolutionary artificial neural network
based on chemical reaction optimization***
J.J.Q. Yu et al.

Chemical reaction optimization (CRO) is a population-based simulated annealing optimization algorithm:

- Molecular structure: solution;
- Potential energy: value of the objective function (to minimize);
- Kinetic energy: tolerance to an increase in energy;
- Decomposition (of a molecule): randomization of 50% of the coordinates;
- Synthesis (collision between molecules): cross-over;
- Motion: random neighbour, mutation.

***Reinforcement learning
neural Turing machines***
W. Zaremba and I. Sutskever

A Neural Turing machine is a Turing machine in which everything (input, output, memory access, etc.) has been replaced by a distribution (over inputs, outputs, possible memory locations, etc.). Reinforcement learning can help train it more efficiently, by avoiding accessing all the memory at each iteration.

Markov logic networks
M. Richardson and P. Domingos

A **Markov logic network** (MLN) is a set of *first order logic* formulas (FOL adds functions and quantifiers to *propositional logic*), with weights.

MLNs extend both FOL and Markov networks. A MLN defines a Markov network: write all the formulas in clausal form (the initial universal quantifiers can be omitted; the existential quantifiers can be replaced by functions; only use disjunctions), e.g.,

Formula	Clausal form
$\forall x F(x)$	$F(x)$
$\forall x \exists y F(x, y)$	$F(x, g(y))$
$\forall x F(x) \Rightarrow G(x)$	$G(x) \vee \neg G(x);$

add a node for each possible grounding of each predicate. The network is large (even if we ensure it is finite by demanding that the values of the functions be among the constants), but inference does not always require grounding the whole network,

DL-Learner manual
J. Lehmann (2014)

Perhaps the only open-source ILP learner.

***Efficient program synthesis using constraint
satisfaction in inductive logic programming***
J. Ahlgren and S.Y. Yuen (2013)

ILP systems often arrange the set of candidate clauses into a lattice, and explore it from the bottom (using A* for Prolog and simulated annealing for Aleph), but many candidates are invalid. Atom is similar, but uses a SAT solver (the DPLL algorithm) to generate valid candidates.

Integer sequence discovery from small graphs
T. Hoppe and A. Petrone (2014)

Encyclopedia of finite graphs: all simple connected graphs with up to 10 vertices and many of their invariants, with the corresponding OEIS sequence.

Tools used: **nauty** (**gen -c** to enumerate the graphs), NetworkX, Graph-tool (C++, with Python bindings, GPL), PuPL, SQLite.

***DeepWalk: online learning
of social representations***
B. Perozzi et al. (2014)

A set of (fixed-length) random walks on a graph can be seen as a corpus: NLP techniques apply (language models, LDA, word2vec, etc.).

***Partial correlation analysis:
applications for financial markets***
D.Y. Kenett et al. (2014)

The influence of a random variable Z on the correlation between two random variables X and Y is

$$d(X, Y : Z) = \rho(X, Y) - \rho(X, Y | Z)$$

where

$$\rho(X, Y | Z) = \frac{\rho(X, Y) - \rho(X, Z)\rho(Y, Z)}{\sqrt{(1 - \rho^2(X, Z))(1 - \rho^2(Y, Z))}}.$$

The average influence $d(X : Z) = \langle d(X, Y : Z) \rangle_{Y \neq X}$ or $d(X) = \langle d(X : Z) \rangle_Z$ is asymptotically Gaussian and can be used in statistical tests.

For two dates, one can compute the rank correlation $\text{Cor}(d_{t_1}(X), d_{t_2}(X))$: the corresponding date×date matrix shows the market stability.

By looking at the influence of stocks Y in a given sector S_i on a stock X , $d(X : S) = \langle d(X : Y) \rangle_{Y \in S}$, one can derive a finer sector decomposition, adapted to companies spanning several sectors.

***A multiple network approach
to corporate governance***
F. Bonacina et al. (2014)

Eigenvalue centrality $x_i = \sum_j A_{ij}x_j$ can be generalized to directed graphs (HITS algorithm, identifying hubs and authorities)

$$a_j = \sum_i A_{ij}h_i$$

$$h_i = \sum_j A_{ij}a_j$$

and directed graphs with coloured topics

$$a_j = \sum_{ik} A_{ijk}h_i t_k$$

$$h_i = \sum_{jj} A_{ijk}a_j t_k$$

$$t_k = \sum_{ij} A_{ijk}a_j h_i.$$

The HITS authority \mathbf{a} and hub \mathbf{h} vectors are the first left and right singular vectors of A , obtained from a low rank decomposition $A \approx \sum_r \sigma_r u^{(r)} \circ v^{(r)}$ or the SVD decomposition $A = UDV'$ – the next singular vectors can often be interpreted as well.

These decompositions can be generalized to tensors: parallel factor analysis $A \approx \sum_r \sigma_r u^{(r)} \circ v^{(r)} \circ w^{(r)}$ or Tucker decomposition $A \approx G \times_1 U \times_2 V \times_3 W$.

In a financial context, one could look at the three following networks:

$$A_{ij1} = \text{percentage of company } i \text{ owned by company } j$$

$$A_{ij2} = \text{number of directors sitting on both boards}$$

$$A_{ij3} = \mathbf{1}_{\text{Cor}(X_i, X_j) > 0.65}.$$

The various singular vectors correspond to various influence networks.

***Inside the network: trading of inside and
outside stocks by corporate insiders***
H. Berkman et al. (2014)

Insiders central in the corporate network earn higher returns when they buy or sell their stock, or nearby stocks.

A network approach to portfolio selection
G. Peralta and A. Zareei (2014)

Stocks with high centrality are more stable; stocks with low centrality bring higher diversification benefits. Avoid stocks with extreme centrality: they are either too volatile or fail to bring any diversification benefit.

***Using friends as sensors
to detect global-scale contagious outbreaks***
M. Garcia-Herranz et al. (2014)

Since the S-shaped cumulative epidemic curve is shifted left for more central nodes, one can build an early-warning system (here, for viral spread of hashtags on Twitter) using a random sample of central nodes. For instance, one could take n nodes at random (control group) and then a random neighbour (followee) of each of them (sensor group – they have higher centrality).

***Common pitfalls
using the normalized compression distance:
what to watch out for in a compressor***
M. Cebrián et al. (2005)

The (incomputable) normalized information distance (NID) is

$$\text{NID}(x, y) = \frac{\text{Max}\{K(x|y) - K(x), K(y|x) - K(y)\}}{\text{Max}\{K(x), K(y)\}}$$

where the conditional Kolmogorov complexity $K(x|y)$ is the length of the shortest program that outputs x when fed y , and $k(x) = k(x|\emptyset)$.

The normalized compression distance is

$$\text{NCD}(x, y) = \frac{\text{Max}\{C(xy) - C(x), C(yx) - C(y)\}}{\text{Max}\{C(x), C(y)\}}$$

where $C(x)$ is the length of x after compression with your favourite algorithm (bzip2 (Burrow-Wheeler transform), gzip (LZ77), PMZZ (Markov model)). To really have a distance, one needs $C(xx) = C(x)$ or, at least, $C(xx) = C(x) + O(\log|x|)$ – but if xx exceeds the block size, this is not guaranteed.

Clustering by compression
R. Cilibrasi and P.M.B. Vitányi (2005)

Initial article on NCD, with applications to clustering.
Implementation: `complearn-gui`.

The Google similarity distance
R. Cilibrasi and P.M.B. Vitányi (2007)

Let $N_{\{x,y\}}$ be the number of web pages containing both x and y , according to Google. Define a probability distribution on all singletons and doubletons of search terms by $g(\{x,y\}) \propto N_{\{x,y\}}$ (allowing $x = y$). The Google code is then $G(x,y) = -\log g(\{x,y\}) = -\log N_{\{x,y\}} + \log N$ and $G(x) = G(x,x)$. The normalized Google distance is another approximation of the NID:

$$\begin{aligned} NGD(x,y) &= \frac{G(x,y) - \text{Min}\{G(x), G(y)\}}{\text{Max}\{G(x), G(y)\}} \\ &= \frac{\log \text{Max}\{N_x, N_y\} - \log N_{xy}}{\log N - \log \text{Min}\{N_x, N_y\}} \end{aligned}$$

We do not know N , but the results do not seem very sensitive to it:

$$\#pages \leq N \leq \#words \text{ per page} \times \#pages.$$

Cross-firm information flows
and the predictability of stock returns
A. Scherbina and B. Schlusche (2013)

Test for Granger causality

$$\begin{aligned} x &\sim \text{lag}(x) + \text{lag}(y) + \text{lag}(\text{market}) \\ y &\sim \text{lag}(x) + \text{lag}(y) + \text{lag}(\text{market}) \end{aligned}$$

for all pairs of stocks on a moving window (12 or 36 months, or 52 weeks) to identify leaders ($p < 5\%$ – you may want to control for multiple testing).

For each stock with leaders, forecast the returns from the past returns of the leaders (using the previous regression – there is one forecast per leader) and invest if most leaders agree.

Beware of non-synchronous trading.

Other signs of leadership include: analyst coverage, institutional ownership, trading volume, increased news coverage (but when the coverage is too high, the company ceases to be a leader: information spreads almost instantly).

Exploring multi-layer flow network
of international trade based on flow distances
B. Shen et al. (2015)

After adding source and sink nodes to the world trade flow network (for some commodity) to ensure that the net flow through each node is zero, one can compute distances:

- Expected first passage time;
- Expected (not only first) passage time (only well-defined for open flow networks, *i.e.*, with source and sink);
- The symmetrization of those, *e.g.*, using a minimum or a harmonic mean:

$$\text{Min}(d_{ij}, d_{ji}), \quad \frac{2d_{ij}^\alpha d_{ji}^\alpha}{d_{ij}^\alpha + d_{ji}^\alpha}.$$

The *trophic level* of a node is the (first passage) distance from the source.

The tone of financial news
and the perceptions of stock and CDS traders
M. Liebmann et al. (2014)

One can estimate the relation between news (they are often already tagged, *e.g.*, “results” or “debt/credit”) and returns (for different assets, *e.g.*, stocks and CDS) using a Naive Bayes classifier, *i.e.*, by looking at the words one by one and keeping the most relevant, *e.g.*,

$$\text{Tone}(w) = \frac{P(+|w)}{P(+)} - \frac{P(-|w)}{P(-)}.$$

One could also use association rule mining [mentioned, not tried – should not work well] or logistic regression [not mentioned – should work with a penalty and word pairs].

Commonly-used lexicons do not work well in a financial context (*e.g.*, “cancer” is usually negative, but neutral in a financial context): one needs a bespoke or dynamic lexicon.

Generalized low-rank models
M. Udell et al. (2014)

This is one of the first papers introducing a statistical algorithm with an implementation, not in R, Python, Matlab or C, but in (both) Julia and Spark.

Principal component analysis (PCA) of $A \in \mathcal{M}_{mn}$,

$$\begin{aligned} (X, Y) &= \underset{\substack{X \in \mathcal{M}_{mk} \\ Y \in \mathcal{M}_{nk}}}{\text{Argmin}} \|A - XY'\|_F^2 \\ &= \underset{X, Y}{\text{Argmin}} \sum_{ij} (A_{ij} - x_i y_j')^2 \end{aligned}$$

can be generalized to

$$\underset{X, Y}{\text{Argmin}} \sum_{(i,j) \in \Omega} \text{Loss}(x_i y_j', A_{ij}) + \sum_i r(x_i) + \sum_j s(y_j).$$

Here are a few examples, for the penalty terms:

- Low-rank matrix completion: $\Omega \subsetneq [1, m] \times [1, n]$;
- Quadratically-penalized PCA: $r = s = \|\cdot\|_2^2$;
- Non-negative matrix factorization: $r(x) = I_{\forall j x_j \leq 0}$ where

$$I_{\text{condition}} = \begin{cases} 0 & \text{if condition is true} \\ \infty & \text{otherwise;} \end{cases}$$

- Sparse PCA: $r(x) = I_{\#\{j: x_j \neq 0\} \leq s}$ or its L^1 relaxation $r = \|\cdot\|_1$;
- Quadratic clustering (k -means): $r(x) = I_{\exists \ell: x = e_\ell}$, where the e_ℓ are the basis vectors;
- Quadratic mixtures: $r(x) = I_{x \in \Delta}$, where Δ is the standard simplex;
- Subspace clustering (approximation of the data as a union of low-dimensional subspaces): partition the columns of x into blocks and set $r(x) = I_{\text{at most one block has nonzero entries}}$;

- Feature selection: $r(x) = \|\cdot\|_2^2$, $s(y) = \|\cdot\|_2$ (not $\|\cdot\|_2^2$), to make y column-sparse (most columns are zero);
- Supervised learning;
- Dictionary learning: $r = \|\cdot\|_1$ to enforce sparsity and $s = \|\cdot\|_2^2$ (to ensure the problem is well-posed when $k \gg m, n$).

Here are a few examples for the loss function:

- Robust PCA:

$$\text{Argmin} \|A - XY'\|_1 + \gamma \|X\|_F^2 + \gamma \|Y\|_F^2;$$

- Weighted PCA: $L_{ij}(u, a) = w_{ij}(u - a)^2$;
- Huber PCA: $L(u, a) = \text{huber}(u - a)$, where

$$\text{huber}(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leq 1 \\ |x| - \frac{1}{2} & \text{if } |x| \geq 1 \end{cases}$$

- Quantile PCA (if the cost of under- and over-estimating A is different: $L_{ij}(u, a) = \lambda(u - 1)_+ + (1 - \lambda)(u - a)_-$;
- Divergences:

$$L_{\text{KL}}(u, a) = a \log \frac{a}{u} - a + u$$

$$L_{\text{IS}}(u, a) = a \log \frac{a}{u} - 1 + \frac{a}{u}$$

$$L_{\beta}(u, a) = \frac{a^{\beta}}{\beta(\beta - 1)} + \frac{u^{\beta}}{\beta} - \frac{au^{\beta-1}}{\beta - 1};$$

- Boolean PCA: $L(u, a) = (1 - ua)_+$ with $a \in \{0, 1\}$ (hinge loss);
- Logistic PCA $L(u, a) = \text{softplus}(-au) = \log(1 + e^{-au})$, with $a \in \{0, 1\}$;
- Poisson PCA $L(u, a) = e^u - au + a \log a - a = L_{\text{KL}}(e^u, a)$, with $a \in \mathbf{N}$ (count data);
- Ordinal PCA

$$L(u, a) = \sum_{b < a} (1 - u + b)_+ + \sum_{b > a} (1 + u - b)_+;$$

- Interval PCA $L(u, [a, b]) = \text{Max}\{(a - u)_+, (b - u)_+\}$;
- Categorical PCA

$$L(u, a) = (1 - u_a)_+ + \sum_{b \neq a} (1 + u_b)_+,$$

where $a \in \llbracket 1, k \rrbracket$ and $u \in \mathbf{R}^k$, i.e., we consider a block of k columns of Y at a time;

- Ordinal PCA $L(u, a) = \sum_b (1 - I_{a > b} u_b)_+$;
- Permutation PCA $L(u, a) = \sum_i (1 - u_{a_i} + u_{a_{i+1}})_+$;
- Ranking PCA $L(u, a) = \sum_{i < j} (1 - u_{a_i} + u_{a_j})_+$.

The problem is not convex, but often *biconvex* (convex in one variable when the other is fixed) and can be solved by *alternating minimization* (alternatively minimize wrt one variable with the other fixed). Each step is easy to parallelize: treat the rows of X separately. We do not need the exact minimum at each iteration: a single step in the right direction is often enough.

To deal with non-finite regularization terms, one can use the *proximal gradient* method:

$$g_1 = \sum_{j: (i, j) \in \Omega} \nabla \text{Loss}(x_i y_j, A_{ij}) y_j$$

$$x_i \leftarrow \text{prox}_{\alpha r}(x_i - \alpha g_i)$$

where $\text{prox}_f(z) = \text{Argmin}_x f(x) + \frac{1}{2}x - z^2$ and $\alpha = 1/k$ (for the k th iteration) or $\alpha = 1/\|g_i\|_2$.

Stochastic gradient can help.

Careful initialization (e.g., SVD, k-means++) is important.

Adaptive subgradient methods for online learning and stochastic optimization **J. Duchi et al. (2010)**

Gradient descent is the iterative algorithm

$$x_{n+1} \leftarrow x_n - \eta g_n,$$

where η is the learning rate and g_n the gradient (or a subgradient) of the loss function.

Projected gradient descent adds a constraint $x \in \mathcal{X}$:

$$x_{n+1} \leftarrow \text{Argmin}_{x \in \mathcal{X}} \|x - (x_n - \eta g_n)\|_2^2.$$

One can give more importance to the directions in which the gradient is larger:

$$x_{n+1} \leftarrow \text{Argmin}_{x \in \mathcal{X}} \|x - (x_n - \eta G_n^{1/2} g_n)\|_{G_n}^2$$

where $G_n = \sum_{1 \leq k \leq n} g_k g_k'$ or $G_n = \sum \text{diag } g_k g_k'$ and $\|x\|_G^2 = \langle x, x \rangle_G = \langle x, Gx \rangle$ defines the Mahalanobis norm.

The proximal gradient is

$$x_{n+1} \leftarrow \text{Argmin}_{x \in \mathcal{X}} \eta \langle g_n, x \rangle + \eta \phi(x) + B_{\psi_n}(x, x_n)$$

where $\psi_n = \psi$ or $\psi_n = n\phi$ or $\psi_n = \sqrt{n}\psi$ or $\psi_n(x) = \|x\|_{\delta I + G_n^{1/2}}$ or $\psi_n(x) = \|x\|_{\delta I + \text{diag } G_n^{1/2}}$ and

$$B_{\psi}(x, y) = \psi x - \psi y - \langle \nabla \psi y, x - y \rangle$$

is the Bregman divergence. The first term pulls x in the direction $-g_n$, as much as allowed by the other terms; the second is a regularization; the third forces x to be close to x_n .

Identifying and attacking the saddle point problem in high-dimensional non-convex optimization **Y.N. Dauphin et al.**

In high dimension, the local minima (of a draw of a Gaussian process) are very close to the global minimum, but there are many critical points far away from the minima (to see it: plot the value of the objective function at critical points versus the proportion of negative eigenvalues). They are surrounded by plateaus, on which gradient methods get stuck. Newton methods are even worse: the saddle points are attractive (Newton methods assume there are no negative eigenvalues and end up going in the wrong direction).

Trust region methods (minimize a first or second order approximation of the objective, with a constraint on the distance from the previous point) fare better. One can “fix” the Newton method for saddle points by replacing the eigenvalues of the Hessian by their absolute values (saddle-free Newton method). This is actually a trust region method, using $|H|$ as a distance, and a first-order approximation. In high dimension, one can stay on a Krylov subspace (a subspace of the form $\text{Span}\{x, Hx, \dots, H^k x\}$, for some x – this approximates the subspace spanned by the most important eigenvectors).

Fast large-scale optimization by unifying stochastic gradient and quasi-Newton methods
J. Sohl-Dickstein et al. (2014)

Combine the mini-batches of stochastic gradient descent with the approximate Hessian (and absence of hyper-parameters) of quasi-Newton methods by keeping track of the (compressed) approximate Hessian of each mini-batch separately

Deep learning with elastic averaging SGD
S. Zhang et al. (2015)

To parallelize stochastic gradient descent (SGD), add an L^2 penalty for the distance to some central, reference value, updated only once in a while – only send updates to the parameter server for parameters that have drifted enough from their initial values.

More precisely, replace the problem

$$\begin{array}{ll} \text{Find} & x \\ \text{To minimize} & f(x) \end{array}$$

with

$$\begin{array}{ll} \text{Find} & x, x_1, \dots, x_n \\ \text{To minimize} & \sum f(x_i) + \rho \sum \|x - x_i\|^2 \\ \text{Such that} & \forall i \ x_i = x. \end{array}$$

For the implementation, ignore the constraint; worker i knows x_i and x , and can compute the contribution of x_i to the gradient; to update x , the workers need to combine their knowledge (this can be done asynchronously).

Bayesian optimization in a billion dimensions via random embeddings
Z. Wang et al.

Bayesian optimization works best in low dimensions: in high dimensions, project on a random subspace – if the intrinsic dimension of the problem is low, this is good enough.

Practical Bayesian optimization of machine learning algorithms
J. Snoek et al.

Bayesian optimization is a class of optimization algorithms that looks for the minimum of an expensive

function; contrary to other optimization algorithms, it keeps (and uses) all the previous function evaluations.

For instance, one can model the function as a *Gaussian process* (GP), often with a *Matern 5/2 kernel* (this makes the function a.s. \mathcal{C}^2 , but not more) and choose the next point to evaluate by maximizing

- The probability of improvement,

$$\text{PI}(x) = P[y(x) \geq \text{best}];$$

- The expected improvement,

$$\text{EI}(x) = E[(y(x) - \text{best})_+];$$

- The upper confidence bound, $\text{LCB}(x) = \mu_x - \kappa\sigma_x$, where $y(x) \sim N(\mu_x, \sigma_x)$.

(Prefer the expected improvement.)

Possible improvements include:

- Modeling the (log-)duration function as well, and minimizing the expected improvement per second;
- Running several function evaluations in parallel and maximizing the expected improvement over all possible outcomes of the pending evaluations.

Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms
J. Bergstra et al. (2013)

If you need to minimize an expensive function, some of whose arguments may be discrete (you need to provide a prior on the parameters, not just bounds). Also check `spearmint`.

Continuous global optimization in R
K.M. Mullen (2014)

Big list of *global* optimization algorithms available in R, *i.e.*, algorithms robust to local minima (simulated annealing, genetic algorithms, particle swarm optimization, branch-and-bound, DIRECT, etc.), with tests on 50 different problems (in the `globalOptTests` package): prefer `nloptr::stogo` (very slow, unless you provide an analytic gradient), `rgenoud::genoud`, `GenSA`.

On best practice optimization methods in R
J.C. Nash (2014)

The `optreplace` package replaces the default optimization algorithms in `optim`:

- `dfoptim::nmkb` instead of Nelder-Mead;
- `Rcgmin` instead of CG;
- `Rvmmmin` instead of BFGS or L-BFGS-B (but it uses a full Hessian, not a low-memory approximation);

`nls` is not replaced, but `nmlrt` is a robust alternative.

Diagnostics are often lacking (there are some in `optimx`).

Stochastic optimization (SANN, DEoptim), constrained optimization (`constrOptim`), fuzzy optimization deserve attention.

Representative/benchmark problems are needed (those in NISTnls/NISTopt are “extreme”).

Convex optimization in R
R. Koenker and I. Mizera (2014)

Long list of applications of convex programming in statistics:

- Median regression;
- Quantile regression;
- Hinge regression (or other piecewise linear convex loss functions);
- Huber M -estimators (the loss function is quadratic when the residuals are small, linear when they are large);
- Penalized quantile regression;
- Support vector machines (hinge loss and L^2 penalty) (**kernlab**);
- Random linear constraints, $P[a'_i x \leq b_i] \geq \eta$, where $a_i \sim N(\alpha_i, \Omega_i)$ are random variables;
- Square root lasso, group lasso;
- Convex regression.

Matrix completion and robust PCA can be expressed with matrix norms such as the spectral norm (largest eigenvalue) or the nuclear norm (sum of the singular values).

Interior point methods usually replace linear inequality constraints with *log-barriers*. For proper cones, one can use a *generalized log* $\lambda : K \rightarrow \mathbf{R}, \mathcal{C}^2$, with $\nabla^2 \lambda \prec_{\text{psd}} 0$ (i.e., concave), such that

$$\exists \theta \quad \forall x \succ_K 0 \quad \forall y \succ_{\mathbf{R}_+} 0 \quad \lambda(yx) = \lambda(x) + \theta \log y.$$

For instance,

$$\begin{aligned} \lambda(x) &= \sum_i \log x_i && \text{for } K = \mathbf{R}_+^n \\ \lambda(x) &= \log(x_0^2 - \|x_{1:n}\|^2) && \text{for } K = \{x \in \mathbf{R}^{n+1} : \|x_{1:n}\| \leq x_0\} \\ \lambda(A) &= \log |A| && \text{for } K = \mathcal{S}_n^+ \end{aligned}$$

Interior point methods work best when the Hessian (of the objective function plus the barrier) is sparse, e.g., when the problem is *additively separable*.

After discretization, infinite mixture models (Kiefer-Wolfowitz)

$$\begin{array}{ll} \text{Find} & f \\ \text{To minimize} & \sum_i \log g_i \\ \text{Such that} & g = Af, \quad f \geq 0, \quad f' \mathbf{1} = 1 \end{array}$$

(where $A_{ij} = \phi(x_i, \theta_j)$, the θ_j are on a fine grid, the x_i are the observations and $\phi(\cdot, \theta_j)$ the probability distribution functions being mixed) or penalized/constrained density estimation problems are convex and additively separable (the penalty could be $\int (\sqrt{f})'$, $\int (\log f)''^2$, $\int |(\log f)''|$; the constraint could be “ f log-concave”). Many of these are implemented in **REBayes**, but it unfortunately relies on Mosek.

trustOptim: an R package for trust region optimization with sparse Hessians
M. Braun (2014)

Trust region optimization minimizes a second order approximation of the objective function in a small region around the current point (it is similar to sequential quadratic optimization, but with constraints, to avoid moving too fast – and it does not misbehave in presence of indefinite Hessians); if the approximation is not good at the new point, or if it does not bring any improvement, shrink the region; if the new point is on the boundary, expand the region.

This is implemented in the **trust** package, but the algorithm does not scale well if the Hessians are dense: **trustOptim** uses the same idea with sparse Hessians (either provide the sparsity structure and use numeric differentiation, or provide an analytic Hessian, e.g., from **CppAD** or **R2admb**).

Square root lasso: pivotal recovery of sparse signals via conic programming
A. Belloni et al.

The square root lasso is

$$\underset{\beta}{\text{Argmin}} \|y - x'\beta\|_2 + \lambda \|\beta\|_1;$$

while the lasso is

$$\underset{\beta}{\text{Argmin}} \|y - x'\beta\|_2^2 + \lambda \|\beta\|_1.$$

Spectral projected gradient methods: review and perspectives
E.G. Birgin (2014)

The spectral projected gradient (SPG, implemented in **BB;** **BBoptim**), combines (projected) gradient descent and secant methods. *Secant methods* approximate the gradient ∇f with an affine function $x \mapsto \phi(x) = a + Bx$, using the last two points,

$$\begin{aligned} \phi(x_{n-1}) &= a + Bx_{n-1} = \nabla f x_{n-1} \\ \phi(x_n) &= a + Bx_n = \nabla f x_n \end{aligned}$$

i.e.,

$$\begin{aligned} B(x_n - x_{n-1}) &= \nabla f x_n - \nabla f x_{n-1} \\ a &= \nabla f x_n - Bx_n \end{aligned}$$

(there are many such functions). The BB method tries to use $B = \sigma I$; this is usually not possible, but the least squares approximation is good enough:

$$\sigma = \frac{\langle \text{previous step, change in gradient} \rangle}{\text{previous step size}}$$

$$\sigma_{n+1} = \frac{s'_n y_n}{s'_n s_n}$$

$$s_n = x_n - x_{n-1}$$

$$y_n = \nabla f x_n - \nabla f x_{n-1}.$$

Spectral methods use this step, multiplied by some coefficient, adjusted to ensure that the objective function improves every M steps (not at every step – it is a non-monotonic method).

What is... a spectrahedron?
C. Vintant

The positive semidefinite cone.

AD model builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models
D.A. Fournier et al. (2011)

Automatic differentiation (AD) differentiates a function written in C++ as follows:

- Evaluate the function and keep track of all the floating point operations performed; if there are loops or conditionals, expand them – the sequence of operations need not always be the same;
- Iteratively compute either $\partial t_j / \partial x_i$ (forward mode) or $\partial f / \partial t_i$ (reverse mode, aka, back-propagation), where the t_i are the intermediate results; reverse mode is more efficient.

ADMB provides a C++-like DSL to specify a function and uses a quasi-Newton algorithm with the gradient from AD to minimize it. It also provides diagnostics useful in statistics: inverse of the information matrix (*i.e.*, variance of the estimated parameters), profile likelihood, MCMC samples from the posterior distribution of the parameters (for the Hessian, they use finite differences: the code generated by AD is often “ludicrously obfuscated” and AD tools “frequently cannot differentiate the code they produce, so you are usually limited to derivatives of order 1”).

Using AD model builder and R together: getting started with the R2admb package
B. Bolker (2013)

If you write down the likelihood of your model yourself, want to use gradient-based optimization methods, but your model is too complex to have its gradient computed by hand or for numeric differentiation to be useful, R2admb can compute the (analytic) gradient for you and optimize the log-likelihood. It generates the usual diagnostics (AIC, vcov, profile likelihood) and (MCMC-based) posterior parameter distributions. It needs ADMB (free, but has to be installed separately) and a C++ compiler, and forces you to write the log-likelihood in C++. There does not seem to be a way of accessing the (compiled, analytic) gradient directly.

Also check glmmADMB – PBSadmb and ADMB2R are older.

In Python, check PyAutoDiff (it uses Theano, and can therefore leverage GPUs).

Getting things in order: an introduction to the R package seriation
M. Hahsler et al.

To reorder the rows and columns of a matrix, e.g. a dissimilarity matrix (in R: `seriation`):

- Try to minimize the number of inversions in each row and column;
- Find a Hamiltonian path on the distance matrix (TSP);
- Minimize the moment of inertia around the diagonal $\sum_{ij} d_{ij} |i - j|$;
- Make the dissimilarities close to the distance to the diagonal, *i.e.*, $d_{ij} \approx |i - j|$ (find $\sigma \in \mathfrak{S}_n$ to minimize the sum of squares $\sum_{ij} (d_{ij} - |i - j|)^2$);
- Ensure similar values are close together by minimizing $\sum_p \sum_{q \in \text{Neigh}(p)} (x_p - x_q)^2$, using 4- or 8-element neighbourhoods;
- Use the first principal component;
- Compute a hierarchical clustering and apply some heuristic to select an ordering compatible with it (in R: `gclus`).

Few of those methods can be generalized to rectangular matrices and, in this case, rows and columns are often treated independently.

Generalized association plots: information visualization via iteratively generated correlation matrices
C.H. Chen

Take the correlation matrix of the columns of a correlation matrix, and iterate.

On the first two principal components, the points form an ellipse; the linear order on this ellipse can be used to solve seriation problems (“rank-2 ellipse seriation”).

After a few more iterations, the matrix often only has ± 1 entries and only two different columns; the observations can be split into two groups; recursively iterating gives a hierarchical clustering. [This assumes the matrix has no symmetries, *i.e.*, its stabilizer for the action $\sigma \cdot A = (A_{\sigma i, \sigma j})_{ij}$ is trivial.]

Package hdlm: regression tables for high-dimensional linear model estimation
T.B. Arnold

It is often claimed that penalized regression does not provide p -values or confidence intervals – but one can use a 2-step approach:

- Fit a *sparse* model (least squares with lasso, quantile regression with lasso, Dantzig selector, etc.) on part of the data (be conservative: we want to keep all the variables in the true model with high probability);
- Fit this smaller model on another (independent) part of the data, and compute p -values and confidence intervals.

The hdlm package also bootstraps the p -values and corrects them (by taking the median).

***Exact and approximate area-proportional
circular Venn and Euler diagrams***
L. Wilkinson (2012)

To plot an Euler diagram, representing the intersections of n sets, with circles:

- Encode all possible intersections as n -bit binary numbers (e.g., $Y_{1101} = X_1 \cap X_3 \cap X_4$); use the same numbering for the disjoint sets, $Z_k = Y_k \setminus \bigcup_{\ell: Y_\ell \subsetneq Y_k} Y_\ell$;
- Define the loss as $\sum (\text{area} - \text{desired area})^2$, where the sum is over the disjoint sets;
- Estimate those areas using either exact computations (compute pairwise intersections, compute the area of the polygon and the lunes), polygon approximations of the circles or a bitmap approach (discretize the circles into pixels);
- For the initial guess, use the desired disk areas and place them using multidimensional scaling (MDS) on the Jaccard distance matrix $d_{ij} = |X_i \cap X_j| / |X_i \cup X_j|$;
- In the optimization, only move the circle centers, first using an (analytical) approximate gradient with a small learning rate, then a numeric gradient.

In R, this is implemented in the `venneuler` package.

Visualizations with Venn and Euler diagrams
L. Micallef

A polygonal Euler diagram can be drawn with a force-directed layout, with attractive and repulsive forces, not only between vertices, but also between edges and polygons.

***Statistical computing in functional data
analysis: the R package fda.usc***

M. Febrero-Bande and M. Oviedo de la Fuente

The `fda.usc` package builds on the other functional data analysis (FDA) packages (`fda`, `ftsa`, `rainbow`, `fpca`, `MFDF`, `fdaMixed`) and provides an abstraction for functional objects (an `fdata` type) aimed at making functional data manipulations as easy as categorical variable manipulations (with the `factor` abstraction). It provides:

- Conversion methods from `fdata`: basis expansion, non-parametric kernel smoothing (`min.basis`, `fdata2fd`, `min.np`);
- L^p distances, semi-metrics (L^p distance between the k th derivatives);
- Depth computations (the average distance between a given observation and all the (other) observations – it is minimum for the median) and the corresponding measures of location and dispersion; they can also be used for outlier detection;
- Smoothed bootstrap;
- Regression with scalar response and functional covariates, with cross-validation and influence measures (Cook’s distance).

***Online prediction under model uncertainty
via dynamic model averaging:
application to a cold rolling mill***
A.E Raftery et al. (2010)

Dynamic model averaging (DMA) generalizes Bayesian model averaging (BMA) to time-varying models. Given n time-varying models (e.g., a time-varying regression, which can be fitted with a Kalman filter), assume that the data may come from a different model at each point in time, and that the correct model changes according to a Markov chain; model the correct model as the hidden state in a hidden Markov model (HMM); use the state probabilities as weights when averaging the n models.

***White noise and simulation
of ordinary Gaussian processes***
B. Puig and F. Poirion

White noise can be defined as a *generalized process* (the stochastic analogue of a tempered distribution), *i.e.*, a continuous linear form on the Schwartz space S (the space of rapidly-decreasing functions ϕ : $\forall \alpha, \beta \sup_x |x^\alpha D^\beta \phi| < \infty$); it is entirely defined by its characteristic function

$$C(\phi) = \int_{\Omega} e^{iX_\phi} dP = \int_{S'} e^{i\langle \omega, \phi \rangle} d\mu(\omega) = \exp -\frac{1}{2} \|\phi\|_H^2.$$

With a stochastic process, we can consider X_t (a random variable) or $X_t(\omega)$ (a number). With a generalized process, we can only look at X_ϕ , where ϕ is a test function, *i.e.*, the weighted average of “ X_t ”, using $(\phi(t))_t$ as weights.

ABC model choice via random forests
P. Pudlo et al. (2014)

ABC posterior probabilities for model selection are not trustworthy. Instead, simulate data from several models, compute a summary statistic, and ask some machine learning algorithms (say, random forests) to predict the model (and its aparmeters) from the summary statistic.

***A survey of metric learning for feature vectors
and structured data***
A. Bellet et al. (2014)

Some machine learning algorithms (k -means, k nearest neighbours) rely on a distance, but choosing it beforehand need not be optimal: *metric learning* uses the data to select the best distance for the problem at hand.

Mahalanobis distance learning looks for a distance of the form

$$d_M(x, y) = \sqrt{(x - y)' M (x - y)}$$

with M positive semidefinite, that optimizes some cri-

terion, e.g. (large merging nearest neighbours, LMNN)

$$\begin{aligned} \text{Find} \quad & M \in \mathcal{S}_+ \\ \text{To minimize} \quad & (1 - \mu) \sum_{(i,j) \in S} d_M^2(x_i, y_i) + \mu \sum_{(i,j,k) \in R} \xi_{ijk} \\ \text{Such that} \quad & \forall (i, j, k) \in R \\ & d_M^2(x_i, x_k) - d_M^2(x_i, x_j) \geq 1 - \xi_{ijk} \end{aligned}$$

where

$$\begin{aligned} S &= \{(i, j) : y_i = y_j \text{ and } x_i \in \text{Neigh}(x_j)\} \\ R &= \{(i, j, k) : (i, j) \in S \text{ and } y_i \neq y_k\} \end{aligned}$$

Regularizations include:

- The mixed $L^{2,1}$ norm, $\|M\|_{2,1} = \sum_i \|M_{i,\cdot}\|_2$, which tends to zero out entire rows of M ;
- Log-det divergence,

$$D(M, M_0) = \text{tr}(MM_0^{-1}) - \log \det(MM_0^{-1}) - d,$$

with $M_0 = I$.

Non-Mahalanobis similarities include:

- $k(x, y) = x'My$, with M not necessarily positive, or even symmetric;
- $k(x, y) = \frac{x'My}{\sqrt{x'Mx}\sqrt{y'My}}$ (generalized cosine similarity);
- Kernalization, kernal PCA (kPCA);
- A neural network (deep belief net, convolutional net) to learn a non-linear projection g so that $\|g(x) - g(y)\|_1$ be small for positive pairs and large for negative pairs.

Local metric learning:

- Cluster the data and learn a different Mahalanobis metric in each cluster; one can also consider the barycenter of Mahalanobis distances;
- The symmetrized Bregman divergence is a smoothly-varying Mahalanobis distance:

$$\begin{aligned} d_\phi(x, y) &= \phi x \phi y - (x - y)' \nabla \phi y \\ d_\phi^{\text{sym}}(x, y) &= d_\phi(x, y) + d_\phi(y, x) \\ &= (x - y)' \nabla^2 \phi(z) (x - y) \text{ for some } z \in [x, y] \end{aligned}$$

One can devise distances for specific types of data:

- Histogram data (points in the standard simplex Δ) and the χ^2 distance

$$\begin{aligned} \chi^2(x, y) &= \frac{1}{2} \sum_i \frac{(x_i - y_i)^2}{x_i + y_i} \\ \chi_L^2(x, y) &= \chi^2(Lx, Ly) \quad (\text{with } L\Delta \subset \Delta) \end{aligned}$$

or the earth mover's distance;

- Strings, trees, graphs with the edit distance.

Handling sparsity via the horseshoe V.M. Carvalho et al. (2009)

To estimate a sparse model, besides the Laplace prior (lasso) and the Student prior (relevance vector machine), consider the *horseshoe prior*

$$\begin{aligned} \beta &\sim N(0, \lambda^2 \tau^2) \\ \lambda &\sim \text{Half-Cauchy} \end{aligned}$$

where λ is the local shrinkage and τ the global shrinkage. The model expects two types of observations: strong signal (no shrinkage) or noise (total shrinkage) – the lasso is more moderate and always has some shrinkage.

Missing value estimation for DNA microarray gene expression data: local least squares imputation H. Kim et al.

To impute missing values in a gene \times experiment matrix (many genes, few experiments):

- Assume we want to impute a value in the first gene (row) g_1 ;
- Find the k closest genes (highest correlation) g_{i_1}, \dots, g_{i_k} ;
- Regress $g_1 \sim g_{i_1} + \dots + g_{i_k}$ and fill in g_1 .

This is a refinement of nearest neighbour imputation, which uses an equal- (or empirically) weighted average of g_{i_1}, \dots, g_{i_k} .

SVD imputation works well for low-noise time series:

- Fill in the missing values with row averages;
- Compute the eigenvectors;
- Regress g_1 against the (first) eigenvectors;
- Predict the missing values;
- Iterate until convergence.

Choquistic regression: generalizing logistic regression using the Choquet integral A. F. Tehrani

A *non-additive* discrete measure (sometimes called a capacity or a fuzzy measure) is a function $\mu : \mathcal{P}(C) \rightarrow [0, 1]$ such that

$$\begin{aligned} \mu(\emptyset) &= 0 \\ \mu(C) &= 1 \\ A \subset B &\implies \mu(A) \leq \mu(B). \end{aligned}$$

It is determined by its *Möbius transform* m :

$$\begin{aligned} \mu(B) &= \sum_{A \subset B} m(A) \\ m(A) &= \sum_{B \subset A} (-1)^{|A|-|B|} \mu(B). \end{aligned}$$

A non-additive measure is k -additive if $|A| > k \implies m(A) = 0$; 0-additive measures are (additive) measures.

The **Choquet integral** generalizes the weighted average (a way of aggregating several criteria)

$$\sum_{c \in C} \mu(\{c\}) f(c)$$

from (additive) measures to non-additive measures. Intuitively, it allows for interactions between the criteria being averaged.

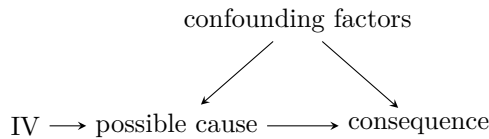
$$\begin{aligned} f : C &\rightarrow \mathbf{R}_+ \\ 0 &\leq f(c_{(1)}) \leq \dots \leq f(c_{(m)}) \\ C_\mu(f) &= \sum_{i=1}^m [f(c_{(i)}) - f(c_{(i-1)})] \mu(\{c_{(1)}, \dots, c_{(m)}\}) \\ &= \sum_{T \subset C} m(T) \times \min_{c \in T} f(c). \end{aligned}$$

In logistic regression, we can replace $w'x$ with $\gamma(C_\mu(x) - \beta)$, where x is seen as a function $i \mapsto x_i$ and μ is k -additive, with k small. The log-likelihood is concave.

Mendelian randomization and causal inference in observational epidemiology N.A. Sheehan et al. (2008)

Mendelian randomization is another name for instrumental variables (IV): find a variable IV so that

- IV $\perp\!\!\!\perp$ confounding factors
- IV \rightarrow possible cause
- IV $\perp\!\!\!\perp$ consequence | possible cause



The IV is often a gene. Gene interactions complicate the picture.

Simulation of fractional Brownian motion T. Dieker (2004)

A stationary stochastic process X is *self-similar* if $X^{(m)} = \frac{1}{m}(X_1 + \dots + X_m)$ and X_1 have the same distribution. It has *long-range dependence* if $\sum \gamma_k$ diverges, where $\gamma_k = \text{Cov}(X_0, X_k)$. A standard *fractional Brownian motion* is characterized by

- $B(t)$ has stationary increments
- $B(0) = 0$, $EB(t) = 0$
- $EB(t)^2 = t^{2H}$
- $B(t)$ is Gaussian.

Its increments $X_k = B(k+1) - B(k)$ are *fractional Gaussian noise*. These are Gaussian processes with covariance functions

$$\begin{aligned} E[B(s)B(t)] &= \frac{1}{2}[t^{2H} + s^{2H} + |t-s|^{2H}] \\ \gamma_k &= E[X_n X_{n+1}] = \frac{1}{2}[|k-1|^{2H} - 2|k|^{2H} + |k+1|^{2H}]. \end{aligned}$$

There is no closed form for the spectral density of fractional Gaussian noise

$$\begin{aligned} f(\lambda) &= \sum_{k \in \mathbf{Z}} e^{ik\lambda} = \\ &2 \sin(\pi H) \Gamma(2H+1) (1 - \cos \lambda) \left(|\lambda|^{-2H-1} + \sum \dots \right) \end{aligned}$$

but approximations are available.

ARFIMA processes, $(1-L)^d X_k \sim \text{ARMA}(p, q)$ also have long memory; the fractional integration is defined by

$$\begin{aligned} (1-L)^d &= \sum_{n \geq 0} \binom{d}{n} (-L)^n \\ \binom{d}{n} &= \frac{\Gamma(d+1)}{\Gamma(d-n+1)\Gamma(n+1)} \end{aligned}$$

The *Hosking method* simulates a general stationary Gaussian process iteratively, in $O(n^2)$: since

$$V_{1:n+1} = \text{Var } X_{1:n+1} = \begin{pmatrix} \gamma_0 & \gamma_1 & \dots & \gamma_{n-1} \\ \gamma_1 & \gamma_0 & \dots & \gamma_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{n-1} & \gamma_{n-2} & \dots & \gamma_0 \end{pmatrix}$$

and $E[X_{1:n+1}] = 0$, we know that $X_{n+1}|X_{1:n} = x_{1:n} \sim N(\mu, v)$, where μ and v are easy to compute (because of the shape of $V_{1:n}$).

The *Choleski method* can simulate arbitrary Gaussian processes; the Choleski matrix can actually be computed explicitly, and this gives an $O(n^3)$ incremental algorithm.

The *Davies and Harte method* embeds the covariance matrix into a circulant matrix

$$\begin{pmatrix} \gamma_0 & \gamma_1 & \dots & \gamma_{n-1} & 0 & \gamma_{n-1} & \dots & \gamma_1 \\ \gamma_1 & \gamma_0 & \dots & \gamma_{n-2} & \gamma_{n-1} & \gamma_0 & \dots & \gamma_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{n-1} & \gamma_{n-2} & \dots & \gamma_1 & \gamma_1 & \gamma_0 & \dots & \gamma_{n-1} \\ 0 & \gamma_{n-1} & \dots & \gamma_1 & \gamma_0 & \gamma_1 & \dots & \gamma_{n-1} \\ \gamma_{n-1} & \gamma_{n-2} & \dots & \gamma_{n-1} & \gamma_1 & \gamma_0 & \dots & \gamma_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_1 & \gamma_0 & \dots & \gamma_{n-1} & \gamma_{n-1} & \gamma_1 & \dots & \gamma_0 \end{pmatrix}$$

which can be efficiently inverted (FFT), to find a square root in $O(n \log n)$.

There are approximate simulation methods (*i.e.*, they sample from a distribution which is not exactly a fractional Brownian motion), using:

- The expression of a fractional Brownian motion as a stochastic integral;
- Queueing theory;
- Random midpoint displacement, *i.e.*, replacing $X_{n+1}|X_{1:n}$ with $X_n|X_{i_1, \dots, i_k}$ in the Hosking or Choleski method;
- Spectral analysis: a stationary Gaussian process can be expressed as a stochastic integral involving its spectral density f ,

$$\begin{aligned} X_n &= \int_0^\pi \sqrt{\frac{f(\lambda)}{\pi}} \cos(n\lambda) dB_1(\lambda) - \\ &\int_0^\pi \sqrt{\frac{f(\lambda)}{\pi}} \sin(n\lambda) dB_2(\lambda); \end{aligned}$$

- Wavelets: the wavelet coefficients of a fractional Brownian motion are almost independent and almost Gaussian, with zero mean and known variance;
- Fractional integration of white noise (white noise is the “derivative” of Brownian motion, seen as a generalized process, aka tempered distribution);
- Series expansions.

The Hurst exponent can be estimated in many ways:

- Aggregated variance $\text{Var } X^{(m)} = m^{2H-2} \text{Var } X$ or absolute moments $E |X^{(m)} - \bar{X}^{(m)}|^k \sim m^{k(H-1)} E |X - EX|^k$ but, in presence of autocorrelation, the finite-sample estimators are biased;
- Higuchi estimator: absolute moment estimator, with $k = 1$ and overlapping windows;
- The slope of the periodogram;
- The variance of the regression residuals, on a moving window (*i.e.*, the aggregated variance, after detrending);
- R/S analysis;
- Likelihood estimator: $(X_1, \dots, X_N) \sim N(0, \Gamma_H)$, where Γ_H is known (as a function of H) – unfortunately, $\det \Gamma_H$ and Γ_H^{-1} are expensive to compute;
- Whittle estimator: compare the periodogram $I(\lambda)$ with the spectral density $f_H(\lambda)$,

$$\hat{H} = \underset{H}{\text{Argmin}} \sum_k \frac{I(\lambda_k)}{f_H(\lambda_k)};$$

- Local Whittle estimator: replace $f_H(\lambda)$ with $C |\lambda|^{1-2H}$;
- Variance of the wavelet coefficients (no detrending is needed if the wavelet has enough vanishing moments).

Speeding up convolutional neural networks using fine-tuned CP-decomposition
V. Lebedev et al. (2015)

In image processing, convolutional neural networks (CNN) perform the convolution of an image with a 4-dimensional tensor:

$$\begin{aligned} x_{ijk} &= \text{pixel at coordinates } (i, j), \text{ in RGB plane } k, \\ &\text{for one of the patches} \\ y_\ell &= b_\ell + \sum_{ijk} a_{ijk\ell} x_{ijk} \\ &= \ell\text{th coordinate of the output, for this patch} \end{aligned}$$

The tensor can be replaced by a low-rank approximation, *e.g.*, the CP-decomposition (a higher-order analogue of PCA).

FitNets: hints for thin deep nets
A. Romero et al. (2015)

To have a small network (*e.g.*, a thin but deep) mimic a large one (wide and deep) do not only learn the outputs, also use the intermediate representations.

Effective use of word order for text categorization with convolutional neural networks
R. Johnson and T. Zhang

Convolutional neural networks (CNN), often used to process images, apply convolutional and pooling layers (one or several pairs) to the data before passing it to a linear classifier.

- A *convolutional layer* takes $k \times k$ (overlapping) regions of the image and gives each of them to m neurons; the weights are shared accross regions; each region, a $3k^2$ -dimensional vector, becomes an m -dimensional vector;
- A *pooling layer* takes the output of a convolutional layer, cuts it into non-overlapping regions, and aggregates them (max-pooling or average pooling), so that each region becomes an m -dimensional vector.

This can also be applied to text:

- Encode length- k regions as bags of vords (*i.e.*, $|V|$ -dimensional vectors) or using indicator variables (*i.e.*, $k|V|$ -dimensional vectors);
- To deal with variable sentence or text length, fix the number of pooling units and dynamically select the pooling region size;
- Use linear rectifiers.

Link analysis, eigenvectors and stability
A.Y. Ng et al.

HITS and PageRank are eigenvalue problems: if the eigengap (the difference between the two largest eigenvalues) is small, the first eigenvector is not stable. HITS is unstable if the maximum degree is large; PageRank is not robust to perturbations to pages with a high score. *Latent semantic indexing* (LSI) (compute the left and right singular vectors of the binary term-document matrix) is a special case of HITS and has the same problem.

Stable algorithms for link analysis
A.Y. Ng et al. (2001)

The *randomized HITS* algorithm is the HITS algorithm, expressed as a random walk with teleportation, à la PageRank (teleportation increases stability).

The *subspace HITS* algorithm uses the first k eigenvectors and defines the scores as

$$a_j = \sum_{i=1}^k f(\lambda_i) \langle e_j, x_i \rangle^2$$

where e_j are the basis vectors (zeroes except in position j), $f(\lambda) = \lambda^2$ (but other weight functions are possible), x_i are the eigenvectors and λ_i the eigenvalues.

Intriguing properties of neural networks
C. Szegedy et al. (2014)

Deep neural networks are discontinuous: it is possible to manufacture an imperceptible alteration of an image to misclassify it.

While it is possible to interpret individual hidden neurons, it is equally easy to interpret random linear combinations of neurons.

***On the effective measure of dimension
in the analysis cospase model***
R. Giryes et al. (2014)

Compressed sensing tries to reconstruct a vector x so that $y = Ax + \text{noise}$, assuming x is sparse.

The cospase model tries to reconstruct a vector x so that $y = Ax + \text{noise}$ assuming Bx is sparse – for instance, x could be an image and Bx its gradient.

***Adaptive compressed sensing for estimation
of structured sparse sets***
R.M. Castro and E. Tanczos (2014)

Compressed sensing, *i.e.*, the reconstruction of a signal from noisy and partial observations, under the assumption that it is sparse, can be generalized to more general supports, e.g., subsets of length s of $\llbracket 1, n \rrbracket$, subintervals of length s of $\llbracket 1, n \rrbracket$, disjoint unions of k such intervals, s -element stars in a graph, etc.

***An exact mapping between the variational
renormalization group and deep learning***
P. Mehta and D.J. Schwab (2014)

The renormalization group is the hierarchical description of a physical phenomenon by progressively integrating out (marginalizing over) small features (“iterative coarse-graining scheme”). It is not unlike deep networks, which integrate small features into larger ones, from layer to layer.

***Taming the monster: a fast and simple
algorithm for contextual bandits***
A. Agarwal et al. (2014)

The ϵ -greedy or epoch-greedy algorithm have a suboptimal regret. Randomized UCB has an optimal regret, but it is complex and slow: it can be simplified and sped up.

***Learning to discover social circles
in ego networks***
J. McAuley and J. Leskovec

A generalization of BigCLAM that allows overlapping or nested communities (not unlike topics) and uses both graph structure and node features.

***Fast, simple and accurate handwritten digit
classification using extreme learning machines
with shaped input-weights***
M.D. McDonnell et al. (2014)

Extreme learning machines (ELM, echo nets – neural nets whose hidden layer is random and not learned) perform as well as deep nets in the digit classification task:

- Augment the training set by distorting it;
- Each hidden unit only operates on a randomly sized and positioned patch of the image.

***Selfieboost: a boosting algorithm
for deep learning***
S. Shalev-Shwartz

Adaboost sequentially fits a model, assigning more weight to currently misclassified observations, and returns a linear combination of the models. Selfieboost only returns the last one; to avoid losing the performance of the earlier models, it adds a penalty to keep the new model forecasts close to the previous ones.

***Move evaluation in Go
using deep convolutional networks***
C. Maddison et al. (2015)

The positions on a Go board are not unlike the pixels of an image: use convolutional neural networks to evaluate a position or a move, using training data from the KGS server

***Teaching deep convolutional neural networks
to play Go***
C. Clark and A. Storkey

More details.

***Efficient gradient-based inference through
transformations between Bayes nets and
Neural nets***
D.P. Kingma and M. Welling (2014)

Bayesian networks and neural networks are the same thing: each can (often) be transformed into the other.

***Hogwild!: a lock-free approach to parallelizing
stochastic gradient descent***
F. Niu et al.

Stochastic gradient descent (SGD) is often easy to parallelize: reformulate the objective function to make it almost separable (a large sum of functions of a small number of variables – that is usually possible if the input is sparse) and forget about concurrency problems (let the processors overwrite each other’s work – it happens too rarely to have a noticeable effect).

***Crypto-nets:
neural networks over encrypted data***
P. Xie et al. (2015)

Homomorphic encryption can be used with neural networks – but since it only provides $+$ and \times , you need to approximate the transfer function with a polynomial (the input space should be compact).

***libDAI: a free and open source C++ library
for discrete approximate inference
in graphical models***
J.M. Mooij (2010)

To estimate Bayesian networks or Markov random fields (MRF). Also check OpenGM2.

***LSH forest: self-tuning indexes
for similarity search***
M. Bawa et al.

Given a locality-sensitive hash (LSH) family $\mathcal{H} = (h_i)_{i \geq 1}$, one usually assigns a fixed-length label $g(p) = (h_1(p), \dots, h_n(p))$ to each point p . Instead, one can use variable-length labels, arranged in a prefix tree, sufficiently long so that all the points have a different label. An LSH forest is a set of such LSH trees.

***A performance evaluation
of open source graph databases***
R. McColl et al. (2014)

Stinger, MTGL, Boost, Giraph and NetworkX perform well.

***Trend filtering methods
for momentum strategies***
B. Bruder et al. (2011)

Here are a few algorithms to identify trends in a time series:

- Moving average, exponential moving average;
- Local regression, which can be seen (for a rectangular kernel) as the discretization of the *Lanczos derivative* (a noise-robust slope)

$$f'(x) = \lim_{\varepsilon \rightarrow 0} \frac{3}{2\varepsilon^2} \int_{-\varepsilon}^{\varepsilon} tf(x+t)dt;$$

- Local polynomial regression, loess smoothing;
- Hodrick-Prescott filter,

$$\hat{y} = \underset{y}{\operatorname{Argmin}} \|y - x\|_2^2 + \lambda \|D^2 y\|_2^2;$$

- Structural model (Kalman filter);
- Low-pass filter, from a Fourier or Wavelet transform;
- Singular spectrum analysis (SSA): let

$$H = \begin{pmatrix} y_1 & \cdots & y_m \\ \vdots & & \vdots \\ y_n & \cdots & y_t \end{pmatrix}$$

and perform a dimension reduction on H ;

- Support vector machines, $Y \sim$ time, with radial basis functions (RBF), and their standard deviation as smoothing parameter;
- Empirical mode decomposition (EMD), Hilbert filter;
- Gaussian processes (not mentioned).

Multivariate filtering methods were listed but not detailed: Kalman filter (with a small number of stochastic components), error correction models, permanent-transitory decomposition.

Chaos in economics and finance
D. Guégan (2009)

Let X_0 be a random variable and $X_n = f(X_{n-1})$ a deterministic dynamic system. To study time series using this model:

- Estimate the embedding dimension;
- Estimate f and f' (using k -NN, radial basis functions, neural nets, etc.);
- Estimate the Lyapunov exponent

$$\lambda = \lim_{n \rightarrow \infty} \frac{1}{n} \sum_{1 \leq t \leq n} \ln |f'(X_t)|,$$

i.e., the speed at which trajectories diverge:
 $\|x_n - y_n\| \approx e^{\lambda n} \|x_0 - y_0\|;$

- Test if it is positive.

***Computing present values:
capital budgeting done correctly***
R. Jarrow (2014)

The value V_t of an asset is related to its expected returns as follows.

$$\begin{aligned} 1 + \mu_{T-1} &:= E_{T-1} \left[\frac{V_T}{V_{T-1}} \right] \\ V_{T-1} &= E_{T-1} \left[\frac{V_T}{1 + \mu_{T-1}} \right] \\ V_{T-2} &= E_{T-2} \left[\frac{V_{T-1}}{1 + \mu_{T-2}} \right] \\ &= E_{T-2} \left[\frac{1}{1 + \mu_{T-2}} E_{T-1} \left[\frac{V_T}{1 + \mu_{T-1}} \right] \right] \\ &= E_{T-2} \left[\frac{V_T}{(1 + \mu_{T-2})(1 + \mu_{T-1})} \right] \\ V_t &= E_t \left[\frac{V_T}{(1 + \mu_t) \cdots (1 + \mu_{T-1})} \right] \end{aligned}$$

(This is the real-world measure, P).

Some textbooks incorrectly price stochastic cash flows as

$$V_0^{\text{wrong}} = \frac{E[V_T]}{(1 + E\mu_0) \cdots (1 + E\mu_{T-1})}.$$

***Correlation in the magnitude
of financial returns***
J. Hämäläinen (2014)

Look at $\operatorname{Cor}(|r_i|, |r_j|)$.

***Understanding the relationship
of momentum with beta***
T. Cenesizoglu et al. (2014)

Beta explains momentum performance.

The stability and accuracy of credit ratings

P. Viegas de Carvalho et al. (2014)

Reratings are not related to changes in the probability of default, but rather to changes in the relative probability of default. Ratings depend on the business or economic cycle.

Data science at the command line

J. Janssens (2014)

Traditional Unix tools (`grep`, `sort`, `uniq`, `head`, `seq`, `paste`, `shuf`, `parallel`, etc.) can be complemented with more recent or specialized tools to process data (download, clean, explore, model) such as `csvkit` (in particular `csvlook` and `csvsql`), `jq` (to transform JSON data, à la XPath), `json2csv`, `scrape` (to transform XML/HTML data, using XPath or CSS selectors), `Rio` (to use R as a filter, to convert a CSV file into another one, or into a plot).

Some of those tools are not standard and were developed by the author for the book. All are available in a (Vagrant) virtual machine.

The book suggests Drake to manage data science workflows and gives a few examples of actual computations, with Tapkee (a C++ library, part of Shogun, for dimension reduction: PCA, MDS, t-SNE, LLE, isomap), Weka, R (via `Rio`), SKLL, BigML – but no mention of Vowpal Wabbit, libsvm, liblinear or libfm. Some traditional and useful Unix commands were also missing: `xargs` (mostly replaced by `parallel`), `dc` (but many prefer `bc`), `file`, `screen`, `tmux`, `csplit` (though `split` was mentioned).

Big data analytics summer school

(Caltech, JPL, Coursera, 2014)

The k -nearest neighbours (kNN) algorithm can be implemented as follows:

- In dimension 1, with binary search;
- In dimensions 2 to 8, with a kd-tree;
- In dimensions greater than 8, with locality-sensitive hashing (LSH) – this only provides approximate neighbours.

In much higher dimensions, kNN no longer works: all the distances are approximately equal (curse of dimensionality).

The Euclidian distance is not a good choice for distance-based algorithms in presence of correlations or differences of scale. One could use the Mahalanobis distance but, for supervised learning, this is suboptimal. For instance, for a classification problem, multi-class discriminant analysis (a form of **metric learning**) looks for the distance that best separates the classes, *i.e.*, maximizes

$$\gamma(V) = \frac{|V'\Sigma_b V|}{|V'\Sigma_w V|}$$

where

V = linear projection of the data

Σ_b = between scatter matrix

Σ_w = within scatter matrix.

The course also covered the following topics:

- Dimension reduction: kNN, feature selection, PCA, metric learning, kernel PCS (kPCA);
- Bootstrap (classical, N out of M , subsampling);
- Visualization;
- Random forests (the sample proximity is the number of trees in which samples i and j end up in the same leaf).

Mining massive datasets

J. Leskovec, A. Rajaraman, J. Ullman
(Stanford, Coursera, 2014)

The complexity of a *map-reduce* algorithm should be measured by the communication cost – that is the main bottleneck, and it is reasonable to assume that all the data has to be moved around. For instance, matrix multiplication is more efficient with a 2-step algorithm: first divide the matrix into blocks (*tiles*) and compute the partial sums, then aggregate them.

Page-rank can be personalized by teleporting to a random page in a set of reference pages (e.g., the DMOZ pages for the topic of interest) instead of a random page; it can be made more robust to spam farms by teleporting to a random page in a set of trusted pages (e.g., edu domains).

Min-hashing is an approximate test for set equality – a **locality-sensitive hashing** (LSH) function for sets. Let $\Omega = \{1, \dots, n\}$ be a set (in practice, it will be a set of words or n -grams); a random permutation $\sigma \in \mathfrak{S}(\Omega)$ defines a (random) hash function

$$h_\sigma(C) = \inf\{i : \sigma(i) \in C\}, \quad C \subset \Omega.$$

The probability that two subsets C_1 and C_2 agree on this hash function is their Jaccard similarity,

$$P[h_\sigma(C_1) = h_\sigma(C_2)] = J(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}.$$

A family H of hash functions $h : S \rightarrow X$, where (S, d) is a metric space and P a probability on H , is said to be (d_1, d_2, p_1, p_2) -sensitive if

$$\forall x, y \in S \quad d(x, y) \leq d_1 \implies P[h(x) = h(y)] \geq p_1$$

$$\forall x, y \in S \quad d(x, y) \geq d_2 \implies P[h(x) = h(y)] \leq p_2.$$

It can be “amplified” by the AND or OR construction, or an AND-OR or OR-AND combination (the distances d_1, d_2 remain the same, probabilities are transformed by $p \mapsto p^r$ or $p \mapsto 1 - (1 - p)^r$).

$$\text{AND} : \forall i, j \in [1, r] \quad h_i(x) = h_j(x)$$

$$\text{OR} : \exists i, j \in [1, r] \quad h_i(x) = h_j(x)$$

Here are some examples of LSH families:

- For the cosine distance, project on a random hyperplane

$$h_v(x) = \text{sign}(v \cdot x), \quad P[h(x) = h(y)] = 1 - \frac{\theta}{\pi};$$

- For the Euclidian distance, project the points on a random line, and put them in buckets of length a .

For approximate text similarity (Jaccard similarity of the bag of words, or the bag of n -grams), one can also look at the following:

- The number of distinct words;
- The N least common words – that gives a lower bound on the distance;
- An index (word, position, number of remaining words), where “position” is the position in the bag of words, sorted with the rarest words first.

The apriori algorithm, for *frequent itemset mining*, proceeds in two steps (for item pairs – n steps for n -item sets): first count the number of occurrences of each item, and pick those that appear at least s times; then, count the pairs of items (but only for those found in the first step), and keep those that appear at least s times. The memory consumption can be reduced:

- PCY: in the first pass, also count pairs, but after hashing them;
- Multihash: idem, with several hash functions;
- Multistage: in pass 1.5, count the hashed pairs, but only for the items from pass 1;
- Toivonen: run the algorithm on a random subset of the data that fits in memory; enlarge the set of candidate itemsets by adding the immediate subsets of those selected by the algorithm; count them (on the whole dataset); enlarge the set of candidate itemsets if necessary; iterate until convergence.

One can model communities using a bipartite graph, with nodes and communities as vertices, and a link probability p_c for each community c ; a link emerges between nodes u and v with probability

$$P(u, v) = 1 - \prod_{c: u, v \in c} (1 - p_c)$$

(or $\text{Max}\{\varepsilon, P(u, v)\}$). The **BigCLAM** model replaces the binary community membership structure with a community strength matrix, $F_{uc} \in [0, 1]$, and a link emerges between u and v with probability

$$P(u, v) = 1 - \prod_c (1 - P_c(u, v)),$$

$$\text{where } P_c(u, v) = 1 - \exp(-F_{uc} \cdot F_{vc}),$$

$$\text{i.e., } P(u, v) = 1 - \exp(-F_u \cdot F'_v).$$

The strength matrix can be estimated as

$$F = \underset{F}{\text{Argmax}} \prod_{(u,v) \in E} P(u, v) \prod_{(u,v) \notin E} (1 - P(u, v))$$

$$= \underset{F}{\text{Argmax}} \sum_{(u,v) \in E} \log \left(1 - e^{-F_u \cdot F'_v} \right) \sum_{(u,v) \notin E} F_u \cdot F'_v.$$

If all the nodes of a graph have the same degree d , $\mathbf{1}$ is an eigenvector of its adjacency matrix with eigenvalue d ; if the graph is not connected, $(1 \cdots 10 \cdots 0)$ and $(0 \cdots 01 \cdots 1)$ are eigenvectors for eigenvalue d ; if there are few edges between the clusters, those vectors are approximate eigenvectors for an eigenvalue close to d . In the general case (**spectral graph partitioning**), consider the Laplacian $L = (\deg G)\text{Id} - A$; the vector $\mathbf{1}$ is an eigenvector with eigenvalue 0; the second smallest eigenvalue is

$$\lambda_2 = \min_x \frac{x' L x}{x' x} = \min_{x' x = 1} \sum_{(i,j) \in E} (x_i - x_j)^2.$$

The coordinates of that vector give the clustering: if there are two clusters, of the same size, they are $\{i : x_i > 0\}$ and $\{i : x_i < 0\}$; in general, look for a jump in $\text{sort}(\mathbf{x})$ and/or look at the third eigenvector.

Trawling finds small communities by looking for complete bipartite subgraphs $K_{s,t}$ – it is a frequent itemset problem.

One can approximately count the number of 1s in a stream of bits, on a moving window of size $k \gg 1$, by keeping track of the counts for blocks of sizes $2^0, 2^1, 2^2, \dots$ and merging them when needed, but the result can be imprecise if the smaller blocks are empty; instead, one can define “size” as being the number of 1s instead of the number of positions.

To estimate the number of different elements in a stream, look at the maximum number R of leading zeroes in a binary hash of those elements (and repeat for many hash functions): 2^R is an estimator of the number of distinct elements but, unfortunately, $E[2^R] = \infty$. This problem can be fixed (*HyperLogLog*).

The singular value decomposition (SVD) gives the best rank- k approximation of a matrix, but it does not preserve sparsity. The **CUR decomposition** addresses this problem:

- C is obtained by taking (say) $4k$ columns of A , at random, sampled with probability $P(j) \propto \|A_{\cdot j}\|_{L^2}^2$ (with replacement): $C = A_{\cdot J}$;
- Similarity, R is obtained from rows of A : $R = A_{I \cdot}$;
- $U = A_{IJ}^\dagger$ (the pseudo-inverse is also computed from the SVD: if $W = XSY'$ is the SVD of W , then $W^\dagger = YS^\dagger X'$, where $S = \text{diag}(s_i)$, $S^\dagger = \text{diag}(s_i^{-1})$ and $s_i^{-1} = 0$ if $s_i = 0$).

Content-based recommendation systems look at the features (e.g., TDIDF, for text) of items highly rated by the user. User-user *collaborative filtering* finds users similar to the target (cosine similarity, or centered cosine similarity aka correlation); item-item collaborative filtering is similar. *Latent factor recommendation systems* decompose the user \times item rating matrix (e.g., SVD with missing values).

There are big-data (1-pass) approximations of the k -means clustering algorithm:

- Read the data in batches; build the clusters progressively; for each cluster, only keep the mean, standard

deviation and count;

- CURE: Run k -means on a random sample of the data; pick 4 points per cluster; move them towards the center; read all the data and assign each observation to the nearest point.

Support vector machines (SVM) are looking for the separating hyperplane with the largest margin

$$\begin{array}{ll} \text{Find} & w, b \\ \text{To maximize} & \min_i (w \cdot x_i + b)y_i \\ \text{Such that} & \|w\| = 1 \end{array}$$

where $y_i \in \{\pm 1\}$. This can be written

$$\begin{array}{ll} \text{Find} & w, b, \gamma \\ \text{To maximize} & \gamma \\ \text{Such that} & \forall i (w \cdot x_i + b)y_i \geq \gamma \\ & \|w\| = 1. \end{array}$$

The *support vectors* are the observations x_i such that the margin $(w \cdot x_i + b)y_i$ is γ . If we remove the constraint $\|w\| = 1$ and impose $(w \cdot x_i + b)y_i = 1$, the margin becomes $\|w\|^{-1}$ (to prove it, consider a support vector x_+ , its symmetrix x_- wrt the hyperplane, and add their margins), and the problem

$$\begin{array}{ll} \text{Find} & w, b \\ \text{To minimize} & \|w\|^2 \\ \text{Such that} & \forall i (w \cdot x_i + b)y_i \geq 1. \end{array}$$

If the data is not separable, add a penalty $c\xi_i$ for each misclassified point

$$\begin{array}{ll} \text{Find} & w, b, \xi \\ \text{To minimize} & \|w\|^2 + c \sum_i \xi_i \\ \text{Such that} & \forall i (w \cdot x_i + b)y_i \geq 1 - \xi_i \\ & \forall i \xi_i \geq 0. \end{array}$$

This can be written with a *hinge loss*

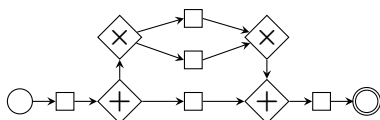
$$\begin{array}{ll} \text{Find} & w, b \\ \text{To minimize} & \|w\|^2 + c \sum_i \text{Max}\{0, 1 - y_i(w \cdot x_i + b)\}. \end{array}$$

The constraints have disappeared: we can use (stochastic) gradient descent.

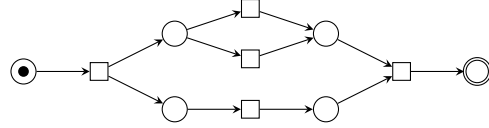
Process mining

W. van der Aalst (Coursera, 2014)

Process mining is the study and modeling of logs (a *log* is a set of traces, a *trace* is a sequence of actions or tasks or activities, with timestamps, and often ancillary data – usually as CSV files, but there is an XML standard: XES). The model is often represented using the business process modeling notation (**BPMN**), with activities \square and decision nodes (XOR split \diamond , XOR join, AND join, \oplus , AND split)



To more easily keep track of the “current state” (the process can be in several branches at the same time), one can use a **Petri net** instead: it is made of transitions \square and places \circ , and the current state is tracked with one or several tokens \bullet : a transition is enabled when there is a token in each of its input places; when it fires, it consumes a token in each of its input places and produces one in each output place (the number of tokens is not constant).



A Petri net is k -bounded if there are at most k tokens in any given place; it is *safe* if it is 1-bounded; a *dead-lock* is a reachable dead marking, *i.e.*, a state in which no transition is enabled; a *live transition* is a transition that can be enabled from any reachable marking; a *workflow net* is a Petri net, with one source place, one sink place, in which all nodes are on a path from source to sink; a workflow net is *sound* if it is safe, has no dead parts, has proper completion (if there is a token in the sink, there are no tokens anywhere else), and has the option to complete.

If x and y are events appearing in the log, $x < y$ means that x is sometimes immediately followed by y . The **causal footprint** of a log is the set (matrix) of relations between events, where the possible relations are:

- $x \rightarrow y$ if $x < y$ and not $y < x$;
- $x \parallel y$ if $x < y$ and $y < x$;
- $x \# y$ if neither $x < y$ nor $y < x$.

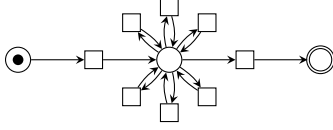
The **alpha algorithm** tries to build a Petri net from the causal footprint of a log. Intuitively, it tries to recognize patterns for XOR/AND split/join (e.g., $a \rightarrow b$, $a \rightarrow c$, $b \# c$ corresponds to a XOR split). More formally:

- L = the set of traces (the log); a trace is a sequence of tasks;
- T = the set of tasks;
- T_i = the set of initial tasks;
- T_o = the set of final (output) tasks;
- X is the set of candidate places, where a candidate place is a pair (A, B) , $A, B \subset T$, such that

$$\begin{array}{lll} \forall a \in A & \forall b \in B & a \rightarrow b \\ \forall a_1, a_2 \in A & & a_1 \# a_2 \\ \forall b_1, b_2 \in B & & b_1 \# b_2 \end{array}$$

- $Y \subset X$ only contains the maximal elements of X ;
- For the set of places $P = Y \cup \{i, o\}$, just add an initial and a final place;
- The set of arcs, F , contains an arc from i to each element of T_i ; an arc from each element of T_o to o ; for places (A, B) , an arc from each element a of A to (A, B) , and an arc from (A, B) to each element b of B .

The algorithm is related to frequent itemset mining. The model learned by the alpha algorithm is not minimal (one can sometimes remove places with no visible effect), and cannot capture loops or non-local dependencies; it is not robust to noise: you can end up with a flower model, which allows any behaviour.



Region-based process discovery tries to address some of those problems. First, define states from the log; for instance, a state could be the trace before a given point, or the trace after a given point, or the whole trace, or the last (or next) k actions (as a sequence, or as a set, or as a multiset); one could also use the other attributes (*resources*) of the events. The states form a transition system. A *region* is a subset of states such that, for each activity a , either a always enters the region, or a always exits the region, or a never enters nor leaves the region. The places are the non-trivial minimal regions.

Language-based regions reduce the problem of finding the places to a linear algebra problem: a place is a solution $(x, y, c) \in \mathbf{N}^3$ of $c\mathbf{1} + Bx - Ay \geq 0$, where A and B are binary matrices, with one column per activity and one row for each prefix of a trace, an element of A is 1 if the activity is in the prefix an element of B is 1 if the activity is in the prefix without its last element; c is the number of tokens, the nonnegativity condition ensures that the number of tokens in a place is never negative, x and y describe the arcs, x is the number of transitions to the place (x, y, c) , y is the number of transitions from the place (x, y, c) .

Inductive process mining decomposes the event log using SEQ, XOR and AND operations. Genetic algorithms are also used in process mining.

To measure the *conformance* of a log and a model:

- Compute the causal footprint of the log and the model, and count the discrepancies;
- Replay the log, add tokens to enable transitions when needed, and count the number of tokens added and the number of tokens remaining at the end (but once the net is flooded with tokens, anything is possible);
- Align the log and the model [no details].

Process models can be *enhanced*:

- At decision points, use statistical models to guess which branch will be taken (data-aware Petri nets);
- Identify bottlenecks and their causes;
- Social networks (for instance, it is easy to identify the structure of an organization);
- Operational support: detect problems, predict the remaining time, predict and recommend the next activity or resource.

MEG decoding using Riemannian geometry and unsupervised classification

A. Barachant

To forecast a binary class $k \in \{0, 1\}$ from multidimensional signals $x_i : [1, N] \rightarrow \mathbf{R}^c$:

- Compute the average for each class, $p_k : [1, N] \rightarrow \mathbf{R}^c$;
- For each class $k \in \{0, 1\}$, combine the signals

$$w_1 p_{k1} + \dots + w_c p_{kc}$$

$$w_1 x_1 + \dots + w_c x_c$$

to maximize the signal-to-noise ratio

$$w^{(k)} = \underset{w}{\operatorname{Argmax}} \frac{\|w' p_k\|_2}{\|w' x\|_2} = \underset{w}{\operatorname{Argmax}} \frac{w' p_k p_k' w}{w' x x' w};$$

- It is an eigenvalue problem: keep the first four eigenvectors w^{k1}, \dots, w^{k4} ;
- As features, use the covariance matrix $\Sigma_i = \frac{1}{N} Z_i Z_i'$ of

$$Z_i = \begin{pmatrix} w^{01'} p_0 \\ \vdots \\ w^{04'} p_0 \\ w^{11'} p_1 \\ \vdots \\ w^{14'} p_1 \\ w^{01'} x_i \\ \vdots \\ w^{14'} x_i \end{pmatrix}$$

- Do not use the variance matrices themselves, but project them on the tangent space (at I , using the logarithm) of the space S_+ of positive definite matrices.

The actual Kaggle task was a form of *transfer learning* (the test samples were not from the same distribution as the training samples – MEGs from a different patient): one can use the forecasts from the trained model as a starting point for the k -means algorithm. But this requires barycenters: in S_+ , the average of a set of points is the point that maximizes the sum of the squared distances, and the distance is

$$d(\Sigma_1, \Sigma_2) = \left\| \log \Sigma_1^{-1/2} \Sigma_2 \Sigma_1^{-1/2} \right\|_F$$

$$= \sqrt{\sum_c \log^2 \lambda_c}$$

where the λ_c are the eigenvalues of $\Sigma_1^{-1/2} \Sigma_2 \Sigma_1^{-1/2}$.

A Riemannian geometry with complete geodesics for the set of positive semidefinite matrices of fixed rank

B. Vandereyken et al. (2013)

It is easy to describe the geodesics of $S_+(n, n)$, the manifold of positive definite matrices of size n , and use them in optimization problems (e.g., line search along a geodesic), but the algorithms do not scale (they are $O(n^3)$). Low-rank approximations are attractive, but the space of positive semidefinite matrices of size n

and rank p , $S_+(n, p)$, has no canonical metric. It can be described as the orbit of

$$\begin{pmatrix} \mathbf{1}_p & \mathbf{0}_{p, n-p} \\ \mathbf{0}_{n-p, p} & \mathbf{0}_{n-p} \end{pmatrix}$$

under the action of GL_n^+ defined by $A \cdot X = AXA^{-1}$. The natural right-invariant metric on GL_n , $g_A(\eta_A, \nu_A) = \text{tr}(A^{-1} \eta'_A \nu_A A^{-1})$, defines a metric on $S_+(p, n)$.

***A differential geometric approach
to the geometric mean
of symmetric positive-definite matrices***
M. Moakher

The matrix exponential $\exp : S(n) \rightarrow S_+(n)$ establishes a bijection between symmetric matrices $S(n)$ and symmetric positive definite matrices $S_+(n) \simeq \text{GL}_n/O_n$.

The geodesic on $S_+(n)$ from I in the direction S is $t \mapsto e^{tS}$.

The geodesic on $S_+(n)$ from P in the direction $S \in S(n) \simeq T_P S_+(n)$ is $t \mapsto P^{1/2} e^{tP^{1/2} S P^{-1/2}} P^{1/2}$.

The distance on $S_+(n)$ is

$$d(P, Q) = \|\text{Log}(P^{-1}Q)\|_F = \left(\sum_{\lambda \in \text{Spec } P^{-1}Q} \ln^2 \lambda \right)^{1/2}$$

(note that $P^{-1}Q$ is not symmetric, but $\text{Spec } P^{-1}Q = \text{Spec } Q^{1/2} P^{-1} Q^{1/2} \subset \mathbf{R}_+^\times$).

The Riemannian mean of positive definite matrices P_1, \dots, P_n , defined as $P = \text{Argmin}_P \sum_k d(P, P_k)^2$, satisfies $\sum_k \text{Log}(P_k^{-1}P) = 0$; it is a generalization of the geometric mean.

***Multiclass brain computer interface
classification by Riemannian geometry***
A. Barachant et al. (2012)

If your data consists of positive definite matrices, rather than arbitrary vectors, algorithms that only use the notions of distance and mean (e.g., kNN, k-means, etc.) are directly applicable using the Riemannian structure of $S_+(n)$. For other algorithms, project the data on the tangent space (at the identity, or any other point, if one makes better sense), using the logarithm.

***Classification of covariance matrices using a
Riemannian-based kernel for BCI applications***
A. Barachant et al. (2013)

More details; algorithm to compute the mean; application to support vector machines.

MEG decoding accross subjects
E. Olivetti et al. (2014)

Transductive transfer learning (TTL):

$$\hat{\theta} = \text{Argmin}_{\theta} \frac{1}{n} \sum_i \frac{P_{\text{target}}(x_i)}{P_{\text{source}}(x_i)} \text{loss}(\theta, x_i, y_i).$$

Factorization machines
S. Rendle

A **factorization machine** is a linear model with interactions,

$$y = \alpha + \sum_i \beta_i x_i + \sum_{ij} \gamma_{ij} x_i x_j$$

where the interaction terms have low rank $\gamma_{ij} = \langle v_i, v_j \rangle$, $v_i \in \mathbf{R}^k$, k small (you can add higher order interactions). It is an alternative to the “hash trick” Vowpal Wabbit uses. It works better than support vector machines for sparse data. (Without α and β , it would just be a matrix factorization.)

For an implementation, check `libfm`.

***Learning recommender systems
with adaptive regularization***
S. Rendle (2012)

Simultaneously learn the model and the regularization parameter:

$$\begin{aligned} \beta_{n+1} &= \underset{\beta}{\text{Argmin}} \text{Loss}(\text{training set}, \beta, \lambda_n) \\ \lambda_{n+1} &= \underset{\lambda}{\text{Argmin}} \text{Loss}(\text{test set}, \beta_{n+1}, \lambda). \end{aligned}$$

(Here, Argmin does not have to be the minimizer: taking a few steps towards the minimum is sufficient.)

***Scaling factorization machines
to relational data***
S. Rendle (2013)

When fitting a model on relational data (often, a star schema, e.g., movie ratings, with movie characteristics (genre, year) and user characteristics (age, gender, friends)), one typically joins (denormalizes) the data. this needlessly inflates the volume of data to process and the running time. Learning on data in normal form is possible (process the fact table; update the weights of the rows on the dimension tables; process the dimension tables).

Tensor decompositions and applications
T.G. Kolda and B.W. Bader (2009)

There are two generalizations of PCA to tensors (using Einstein’s summation convention):

$$\begin{aligned} \text{CP:} \quad & x_{ijk} = g^r a_{ir} b_{jr} c_{kr} \\ \text{Tucker:} \quad & x_{ijk} = g^{pqr} a_{ip} b_{jq} c_{kr} \end{aligned}$$

The CP (canonical decomposition/parallel factors) expresses the tensor x as a sum of rank-1 tensors. The *rank* of x is the minimal number of terms in such a sum. Contrary to the notion of rank for matrices:

– The ranks over \mathbf{R} and \mathbf{C} may be different;

- Among tensors of a given shape, the typical rank (any rank that occurs on a set of non-zero measure) and the maximum rank can differ;
- “The” typical rank is not unique;
- The best rank-1 approximation need not be part of a best rank-2 approximation;
- The best rank- k approximation need not exist: the set of rank- k tensors is not closed, *i.e.*, some tensors can be approximated arbitrarily closely by a tensor of smaller rank – the *border rank* of x is the minimum number of rank-1 tensors needed to approximate x with an arbitrarily small error.

The CP decomposition can be estimated using alternating least squares (ALS).

Probabilistic principal component analysis
M.E. Tipping and C.M Bishop (1999)

PCA is a limiting case of (isotropic) factor analysis, $X \sim N(\mu, WW' + \sigma^2 I)$, when $\sigma^2 \rightarrow 0$ (σ^2 is the average variance due to discarded components). PPCA allows for missing values.

In R, check the `pcaMethods` package (for probabilistic and bayesian PCA).

NICE: non-linear independent component estimation
L. Dinh et al. (2014)

Given a random variable $X : \Omega \rightarrow \mathbf{R}^n$, a non-linear independent component decomposition is a transformation $f : \mathbf{R}^n \rightarrow \mathbf{R}^n$ such that the density of $H = f(X)$ factorizes: $p_H(h) = \prod_d p_{H_d}(h_d)$.

It can be estimated by maximum likelihood, $p_X(x) = p_H(f(x)) |\det f'|$, *i.e.*,

$$\text{Argmax}_{f, p_{H_1}, \dots, p_{H_n}} \sum_x \sum_d \log p_{H_d} f_d x + \log |\det f'|.$$

We look for a transformation (built from three layers of building blocks) of the form

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= x_2 + m(x_1) \end{aligned}$$

which ensures that $|\det f'|$ and f^{-1} are easy to compute.

Sparse matrix factorization
B. Neyshabur and R. Panigrahy (2014)

The sparse matrix factorization problem aims to write a matrix Y as a product of sparse matrices $Y = X_1 X_2 \dots X_n$ – PCA can be seen as a special case, where X_1 has few columns and X_2 few rows.

Assuming the matrices are square, random, centralized, normalized, and the coefficients of Y are 0, 1 or -1 , $X_1 X_1' = \text{round}(Y Y')$ with high probability; X_1 can be recovered from $X_1 X_1'$, and one can then iterate.

Blind denoising with random greedy pursuits
M. Moussallam et al. (2014)

To find a sparse reconstruction of a time series y , choose “atoms” (elementary time series $\Phi = (\phi_1 | \dots | \phi_n)$) and fit $y = \Phi \alpha + w$, where α is sparse:

$$\begin{aligned} &\text{Find} && \alpha \\ &\text{To minimize} && \|\alpha\|_0 \\ &\text{Such that} && \|y - \Phi \alpha\| \leq \varepsilon. \end{aligned}$$

One can use a greedy algorithm (matching pursuit, MP) to find an approximate solution – for the stopping criterion, look at

$$\text{Max}_{\phi} \frac{|\langle y, \phi \rangle|}{\|y\|_2 \|\phi\|_2}.$$

Instead of a single sparse representation, one can compute several (as with random forests: at each step, choose a new atom in a random subset of Φ) and average them.

Dimensionality reduction for k -means clustering and low rank approximation
M. Cohen et al. (2014)

To speed up approximate k -means or PCA, use a sketch of the data (row and column sampling, approximate SVD (CUR), Johnson-Lindenstrauss projection).

Alternating least squares for personalized ranking
G. Takács and D. Tikk (2012)

Alternating least squares (ALS) are often used to find $\hat{R} = P'Q$, with P and Q rectangular with a small number of columns, to minimize

$$\sum_{u \in \text{Users}} \sum_{i \in \text{Items}} c_{ui} (\hat{r}_{ui} - r_{ui})^2.$$

It can also be used with a ranking objective

$$\sum_{uij} c_{ui} s_j ((\hat{r}_{ui} - \hat{r}_{uj}) - (r_{ui} - r_{uj}))^2.$$

HyperLogLog in practice: algorithmic engineering of a state of the art cardinality estimation algorithm
S. Heule et al. (2013)

HyperLogLog is biased for small counts: one can (empirically) estimate the bias and correct it. (HyperLogLog++ introduces a few more changes, mostly to reduce memory usage.)

***In-code computation
of geometric centralities with HyperBall:
hundred billion nodes and beyond***
P. Boldi and S. Vigna

To count the number $|B(x, r)|$ of nodes in the ball of center x and radius r in a graph, use a HyperLogLog counter c_v for each node and estimate $B(x, r+1)$ from $B(x, r)$.

This can be used to compute the *harmonic centrality*

$$\text{centrality}(x) = \sum_y \frac{1}{d(x, y)} = \sum_{r \geq 1} \frac{|B(x, r)|}{r}.$$

The power of comparative reasoning
J. Yagnik et al.

The winner-takes-all (WTA) locality-sensitive hash (LSH) is

$$\begin{cases} \mathbf{R}^n & \longrightarrow \{1, \dots, k\} \\ \mathbf{x} & \longmapsto \underset{1 \leq j \leq k}{\text{Argmax}} x_{\sigma(j)} \end{cases}$$

for a random permutation $\sigma \in \mathfrak{S}_n$.

By using several permutations σ and choosing $k = 2$, we get a binary hash.

One can apply a kernel before computing the hash.

***OpenGM: a C++ library
for discrete graphical models***
B. Andres et al. (2012)

C++ template library implementing algorithms for discrete graphical models (loopy belief propagation, sequential tree reweighted belief propagation, graph cut, etc.), *i.e.*, given a factor graph and a semiring $(\Omega, \otimes, 1, \oplus, 0)$, computing $\bigoplus_x \bigodot_F \phi_F(x_{\text{neigh}(x)})$; examples include

Optimization	$(\mathbf{R}, +, 0, \min, \infty)$
Marginalization	$(\mathbf{R}^+, \cdot, 1, +, 0)$
Constraint satisfaction	$(\{0, 1\}, \wedge, 1, \vee, 0)$

***MPPack:
a scalable C++ machine learning library***
R.R Curtin et al. (2012)

An LGPL C++ template machine learning library (also check Weka, Shogun, mlpy, sklearn, MLLib).

Practically accurate floating point math
N. Toronto and J. McCarthy (2014)

Measure the floating point error when approximating a real number r with a floating point number x as

$$\text{error}(x, r) = \frac{x - r}{\text{ulp}(\text{fl}(r))}$$

where

$\text{ulp}(y)$ = distance from y to the next float

$\text{fl}(r)$ = float closest to r

$\text{ord}(x)$ = (signed) number of floats between 0 and x

$\text{ord}^{-1}(n)$ (needed to compute ulp).

It is the number of floats between x and r .

When implementing a numeric function, check where it is ill-conditioned (“the badlands”), e.g., for functions $f : \mathbf{R} \rightarrow \mathbf{R}$, check where the condition number $|xf'(x)/f(x)|$ exceeds 4.

The examples use typed/racket (for the plots as well) and MPFR.

***Fast approximate nearest neighbors
with automatic algorithm configuration***
M. Muja and D.G. Lowe

Comparison of randomized kd-trees and hierarchical k -means for fast approximate nearest neighbours (FANN).

***CRDTs: consistency
without concurrency control***
M. Leřia et al. (2009)

CRDTs are data structures whose operations commute: they can be used in a distributed environment. For instance:

- A set with a single add-element operation;
- A set with add and delete operations, if no element is ever added twice: keep two lists, one for added elements, one for deleted elements, and clean them from time to time;
- An ordered set, with insert-at and delete operations: use a dense index space (not \mathbf{R} , but paths in a binary tree).

***A comprehensive study of convergent
and commutative replicated data types***
M. Shapiro et al. (2011)

Eventual consistency, in a distributed/asynchronous environment, can be guaranteed by the data structures themselves, without synchronization. For instance:

- Add-only set;
- Distributed maximum (when the maximum changes, broadcast it to the other replicas);
- Counter: broadcast +1 or −1 – the reception order is irrelevant;
- Register: store value-timestamp pairs and only keep the value with the latest timestamp;
- Set with add and remove operations, if removed elements cannot be added back: use two add-only sets, for elements added and elements removed, and occasionally garbage-collect them;
- Set with add and remove operations, in which add has priority over remove in case of a conflict (e.g.,

shopping cart): when adding an element, also store a unique identifier (a new one, if the element was already there); remove removes the elements with the identifiers known to the replica that received the order;

- Graphs: like sets;
- Directed acyclic graphs: that is a global property; it cannot be ensured in general without synchronization.
- Partial order: like sets;
- Total order (e.g., collaborative editing): either add an `addRight(position,content)` operation, which stores the content-timestamp pair in a linked list or an `addBetween` operation, for which the total order is defined by a dense set of identifiers (trees: Logoot or Treedoc – but they have to be rebalanced (garbaged-collected) from time to time).

Generation and analysis of constrained random sampling patterns
J. Pierzchlewski and T. Arildsen

Random sampling of analog (1-dimensional) signals is preferable to regular sampling. Simple sampling patterns include: a regular grid with noise, or adding noise to the previous point.

The algorithmic foundations of differential privacy
C. Dwork and A. Roth (2014)

To ensure plausible deniability, use randomized algorithms (e.g., answer truthfully with probability $\frac{1}{2}$, answer randomly otherwise – for real-valued functions, perturb the result with the Laplace distribution). A randomized algorithm \mathcal{M} is differentially private if

$$\|x - y\| \leq 1 \implies P[\mathcal{M}(x) \in S] \leq e^\epsilon P[\mathcal{M}(y) \in S] + \delta.$$

The privacy loss is $\ln \frac{P[\mathcal{M}(x) = \xi]}{P[\mathcal{M}(y) = \xi]}.$

When answering several queries, the system should refuse to answer similar ones (e.g., the number of people with disease D, and the number of people not named X with disease D).

Computing arbitrary functions of encrypted data
C. Gentry (2008)

The following encryption scheme, which adds noise to the cleartext, is somewhat homomorphic – the noise progressively increases and eventually becomes too large.

cleartext: $m \in \{0, 1\}$
 key: $p \in \{0, 1\}^P, p \equiv 1 \pmod{2}$
 ciphertext: $c = m' + pq$
 where $m' \in \{0, 1\}^N, m' \equiv m \pmod{2}$
 $q \in \{0, 1\}^Q$
 somewhat preserved: $+, -, \times$

To make it homomorphic, try to find functions $f_{i \rightarrow i+1}$ that transform $\text{Encrypt}(p_i, m)$ to $\text{Encrypt}(p_{i+1}, m)$ (i.e., that change the key and reset the noise), using the somewhat preserved operations (this is possible after a small change in the encryption scheme). This is slow, there is no division, and no inequalities.

A comparative study of discretization methods for naive Bayes classifiers
Y. Yang and G.I. Webb (2002)

When discretizing data, use \sqrt{n} quantile bins, but make sure each bin has at least 30 observations.

The Winograd schema challenge
H.J. Levesque et al. (2012)

A replacement for captchas (or the Turing test): resolve referential ambiguity (e.g., “the trophy does not fit in the suitcase because it is too big”, what is too big: the trophy or the suitcase?).

Sloane’s gap: do mathematical and social factors explain the distribution of numbers in the OEIS?
N. Gauvrit et al. (2011)

Plotting the number of occurrences of an integer in the OEIS shows two clouds, with a similar exponential decay, and a gap inbetween. Computational complexity explains the decay, but not the gap – it could be due to social factors.

Profiling R on a contemporary processor
S. Sridharan and J.M. Patel (2014)

R is slow because of cache misses, caused by:

- Pointer-chasing the the garbage collector (GC);
- Linear algebra operations: R should provide both row- and column-major storage and/or use blocking;
- Custom C code in packages (e.g., `rpart`): it should be cache-conscious.

The GC problems are made worse by:

- Unnecessary attributes (e.g., row names, which are automatically added);
- Unnecessary copies; intermediate object creation (arithmetic operations on vectors or matrices);
- Inefficient implementations (e.g., subsetting with a boolean vector recomputes the row indices for each column).

Convex optimization in Julia
M. Udell et al. (2014)

The `Convex.jl` package brings disciplined convex programming (DCP) to Julia.

Besides the positive orthant, the second order cone, and the semi-definite cone, note the *exponential cone*,

$$\{(x, y, z) \in \mathbf{R}^3 : y \geq 0, ye^{x/y} \leq z\},$$

used for the convex functions $\exp(x)$, $-\log(x)$, $\log \sum_i \exp x_i$.

Computing in operations research using Julia
M. Lubin and I. Dunning (2013)

JuMP.jl relies on macros rather than operator overloading (as in CVX, Convex.jl, PuPL, Pyomo, etc.) and can therefore efficiently build larger problems.

A review of goal programming for portfolio selection
R. Azmi and M. Tamiz (2010)

“Goal programming” is a vague notion that refers to the various ways of selecting a single solution of a multi-objective optimization problem, “find x to minimize $g_1(x), \dots, g_n(x)$ ” somewhere on the efficient frontier. For instance:

- Minimize $\sum_i \lambda_i g_i(x)$;
- Minimize $\text{Max}_i g_i(x)$;
- Minimize $g_1(x)$; among the solutions, minimize g_2 ; continue until there is only one solution left;
- etc.

Portfolio selection with higher moments: a polynomial goal programming approach to ISE30 index
G. Kemalbay et al. (2011)

Indicator-based selection in multiobjective search
E. Zitzler and S. Künzli (2004)

Multiobjective search tries to find a good approximation of the Pareto set – a good solution is close to the Pareto set and diversified. These notions are rarely clearly defined, but could be formalized, e.g., one could try to maximize the volume dominated by the solution. Here is one algorithm:

- Start with a large candidate set;
- Compute the “fitness” of each point in it (relative to the others);
- Remove the worst point; update the fitness of the others;
- Iterate (until some termination criterion is met, e.g., the population size).

The *indicator-based evolutionary algorithm* (IBEA) defines the fitness of a point $a \in A$ as

$$F(a) = \sum_{b \in A \setminus \{a\}} -\exp -I(\{a\}, \{b\})/\kappa$$

where κ is a parameter (the temperature) and I is a *binary quality indicator*, e.g.,

$$I(A, B) = \text{Min}\{\varepsilon : \forall b \in B \quad \exists a \in A \quad \exists i \in \llbracket 1, n \rrbracket \\ f_i(a) - \varepsilon \leq f_i(b)\}$$

or

$$I(A, B) = \begin{cases} I(B) - I(A) & \text{if } \forall b \in B \exists a \in A \ a \succ b \\ I(A \cup B) - I(A) & \text{otherwise} \end{cases}$$

where $I(A)$ is the volume dominated by A .

Sparse and stable Markowitz portfolios
J. Brodie et al. (2008)

For sparse portfolios, add an L^1 penalty to the portfolio optimization problem.

Sparse portfolio selection via quasi-norm regularization
C. Chen et al. (2013)

For sparse portfolios, add an L^p penalty ($0 < p < 1$) to the portfolio optimization problem; it is no longer convex, but can still be solved in polynomial time via interior point methods.

Optimal risk budgeting under a finite investment horizon
M. López de prado, R. Vince and Q. Zhu (2013)

To decide which proportion x_i of one’s wealth to invest in asset i , one can use $\hat{x} = \text{Argmax}_x f_{Q,U}(x)$, where $f_{Q,U}(x)$ is the expected utility for utility function U after Q periods. The Kelly principle is $\hat{x} = \text{Argmax}_x f_{\infty, \text{linear}}(x)$. The *leverage space* seems to be the graph of $x \mapsto f_{Q, \text{linear}}(x)$.

A factor model for non-linear dependences in stock returns
R. Chicheportiche and J.P. Bouchaud (2013)

The *non-linear correlation* between two random variables X and Y generalizes $\text{Cor}(|X|, |Y|)$ and $\text{Cor}(X^2, Y^2)$:

$$C_p(X, Y) = \frac{1}{p^2} \log \frac{E[|XY|^p]}{E[|X|^p]E[|Y|^p]}.$$

In the standard factor model $x_i = \sum_k \beta_{ki} f_k + e_i$, the non-linear correlations $C_p(f_k, f_\ell)$, $C_p(f_k, e_i)$, $C_p(e_i, e_j)$ are non-zero. The nested factor model, for the volatilities of the factors f and the residuals e , with two factors, Ω_0 and Ω_1 , can account for them:

$$f_k = \varepsilon_k \exp[A_{k0}\Omega_0 + A_{k1}\Omega_1 + \omega_k] \\ e_j = \eta_j \exp[B_{j0}\Omega_0 + B_{j1}\Omega_1 + \varpi_k].$$

Commodity futures and market efficiency
L. Kristoufek and M. Vosvrda (2013)

To measure the efficiency of a given market (or asset), average the following measures:

- Hurst exponent (long-term efficiency): local Whittle estimator, GPH estimator;
- Fractal dimension (short-term): Hall-Wood estimator, Genton estimator;
- Entropy: Pincus approximate entropy.

The article also gives a concise review of those estimators.

Implied Hurst exponent and fractional implied volatility: a variance term structure model
K.Q. Li and R. Chen (2014)

One can compute the Hurst exponent from option prices:

- Using the fractional Black-Scholes formula; a single option does not give (σ, H) , but just $V = \sigma^2 T^{2H}$; use several maturities and regress $\log V \sim \log T$;
- Using a model-free approach, very similar to the VIX computations, to compute V .

Forecasting the NOK/USD exchange rate with machine learning techniques
T. Papadimitriou et al.

One can forecast the exchange rate

$$FX_{t+1} \sim FX_t + (M_2 - M_2^{\text{US}}) + (\text{GDP} - \text{GDP}^{\text{US}}) + (\text{IR} - \text{IR}^{\text{US}}) + (\text{Inflation} - \text{Inflation}^{\text{US}}) + \text{Oil} + t,$$

where the inflation is a forecast from an ARMA model or from the forward rate, using support vector regression

$$\hat{\beta} = \underset{\beta}{\text{Argmin}} \frac{1}{2} \|\beta\|^2 + C \sum_i \phi(y_i - \beta' x_i)$$

$$\phi = \begin{array}{c} \diagup \quad \diagdown \\ | \end{array}$$

(with the kernel trick). The resulting model can be made more interpretable using a “dynamic evolution neuro-fuzzy inference system” (DENFIS), *i.e.*, a set of fuzzy rules of the form “Rule m : if x_1 is R_{m1} and ... and x_n is R_{mn} then $y = \sum_i \beta_{mi} x_i$ ”, where R_{m1} is a Gaussian membership function

$$R_{m1} \propto \exp - \frac{(x - c)^2}{2\sigma^2}.$$

Rationality of survival, market segmentation and the equity premium puzzle
Y. Gabovich (2014)

A supply-and-demand model, with young people investing in equities and retirees in bonds, can explain the equity premium puzzle (the difference in returns between bonds and equities is too large to be explained by risk aversion alone).

Do patented innovations affect cost of equity capital?
S.P. Hegde and D.R. Mishra (2014)

Cost of equity increases with R&D expenses but decreases with patents: they reduce the uncertainty about the marketability of the research.

Say it again Sam: the idiosyncratic information content of corporate conference calls
J. Cicon (2014)

To measure the information content of corporate conference calls, compare the two parts of the call (management presentation and Q&A session), *e.g.*, with cosine similarity, to estimate how much unscripted information is provided.

Management forecasts and the cost of equity capital: international evidence
Y. Cao et al. (2014)

The presence of management forecasts lowers the cost of equity.

Do social firms catch the drift? Social media and earnings news
V. Bhagwat and T.R. Burch (2013)

Twitter firms (more than 0.1 tweet per day, on average, since the creation of the twitter account – the mapping from firm to twitter account was done manually) have stronger post-earnings announcement drift (PEAD).

A general option valuation approach to discount for lack of marketability
R. Brooks (2014)

To price a non-marketable asset, one can model the price of the corresponding marketable asset as the sum of the unmarketable price and an option, *e.g.*, an option to sell the asset at the current market price, or at the best price in the period (if the investor has market timing abilities) or at the average price in the period (European put, lookback put, average-strike put).

Stock market ambiguity and the equity premium
P.C. Andreou et al. (2014)

Ambiguity (uncertainty about the probability distribution of future assets) can be measured as the dispersion (standard deviation or mean absolute deviation) of the volume-weighted strike prices of S&P 500 index options. It is a better predictor of future returns than other ambiguity proxies (analysts' forecasts, stock turnover, press) or other option-implied measures (slope of the volatility smirk, risk-neutral variance, skewness or kurtosis, hedging pressure, *i.e.*, OTM puts / ATM calls), and almost as good as the variance risk premium, $(VRP = \text{Var}^Q X_t - \text{Var}^P X_t)$, where $\text{Var}^Q X_t$ is computed from option prices and $\text{Var}^P X_t = \text{Var}^{\text{realized}} X_{t-1}$.

Credit risk models II: structural models
A. Elizalde (2005)

Structural default models model the value of the assets of a company as a geometric Brownian motion; a default occurs when the value of the assets is below some threshold

- at the end of the period;
- at any point during the period;
- or on some sufficiently long subinterval of the forecast period.

These correspond to the Merton model (European options), first passage models (FPM, barrier options) and liquidation process models. Those models do not work well.

To account for correlation and contagion, one can model the value of the assets (of several companies) as diffusions with correlated innovations and correlated jumps.

***Asymmetric dependence, tail dependence
and the time interval
over which the variables are measured***
B.U. Kang and G. Kim (2014)

Look at the asymptotic behaviour of the n -period copula:

$$\left| \lambda^{(n)}(q) - \lambda^{(n)}(1-q) \right| \xrightarrow{n \rightarrow \infty} 0$$

$$\lambda_U^{(n)}, \lambda_L^{(n)} \xrightarrow{n \rightarrow \infty} 0$$

where $\lambda^{(n)}(q)$ is the n -period quantile and $\lambda_U^{(n)}, \lambda_L^{(n)}$ the upper and lower tail dependence indices.

Stock and index derivatives and markets
J. Spina (2014)

The main option strategies are:

- Dividend spread: profit from investors that fail from exercising their American options in time;
- If you own stocks of your own firm, want to keep the dividends and voting rights, but want some protection: buy OTM puts (you may also want to sell OTM calls to finance the puts);
- Variable prepaid forward (sell the stock you own in the future, but get the money now);
- If you want to bet on the performance of a stock (e.g., your firm), use an option collar, sell ATM puts, buy ATM calls.

Relative alpha
J.C. Jackwerth and A. Slavutskaya

The relative alpha of a hedge fund is the excess performance wrt its peers, where the peers are funds for which the return difference has a small variance. More formally,

$$\alpha_i = \sum_{j \neq i} w_{ij} E[r_i - r_j]$$

$$w_{ij} \propto k(h^{-1} \text{Var}[r_i - r_j])$$

k : kernel

h : bandwidth (Silverman's rule of thumb).

Factor high-frequency based volatility models
K. Sheppard and W. Xu (2014)

Factor model to estimate large realized covariance matrices.

***A multivariate model
of strategic asset allocation with longevity risk***
E. Bisetti et al. (2014)

If you use reinsurance as a cheap source of leverage (like Berkshire), do not forget to account for longevity risk – it is negligible for short-term investors, but significant for long-term ones.

***Consumption-based asset pricing
with rare disaster risk***
J. Grammig and J. Sönksen (2014)

High equity risk premium may be due to fears of rare disasters.

***Optimal hedging
with the vector autoregressive model***
L.T. Gatarek and S. Johansen (2014)

With several assets (3+) and cointegration relations (2+), use portfolio construction tools (e.g., minimum variance) on the set of stationary portfolios.

***Classifying as defensive or cyclical
a bivariate wavelet analysis perspective***
J. Bruzda

Given two signals x and y , compute the non-decimated Hilbert coefficients u and v and define

$$\text{Cov}_j(x_t, y_{t+\tau}) = u_{jt} \bar{v}_{j,t+\tau}$$

$$\text{Var}_j x_t = \text{Cov}_j(x_t, x_t)$$

$\text{Re Cov}_j(x_t, y_t)$: wavelet cospectrum

$\text{Im Cov}_j(x_t, y_t)$: wavelet quadrature spectrum

$$\text{Cor}_j(x_t, y_t) = \frac{\text{Cov}_j(x_t, y_t)}{\sqrt{\text{Var}_j x_t \text{Var}_j y_t}}$$

$|\text{Cor}_j(x_t, y_t)|^2$: wavelet coherence

$|\text{Cor}_j(x_t, y_t)|$: wavelet coherency

$\theta_j(x_t, y_t) = \arg \text{Cov}_j(x_t, y_t)$ phase spectrum

$$\tau_j(x_t, y_t) = -\theta_j(x_t, y_t) \cdot \frac{2^{j+2}}{6\pi} \quad \text{time delay}$$

$$\beta_j(x_t, y_t) = \frac{\text{Cov}_j(x_t, y_t)}{\text{Var}_j x_t}$$

$|\beta_j(x_t, y_t)|$: wavelet gain.

In R, check `waveslim::modwt.hilbert(x, "k313", 4)`.

Optimality of momentum and reversal
X.Z. He et al. (2014)

One can add momentum and reversal to a geometric

Brownian motion

$$\frac{dS}{S} = [\phi m_t + (1 - \phi) \mu_t] dt + \sigma_S dZ_S$$

$$d\mu_t = \alpha(\bar{\mu} - \mu_t)dt + \sigma_\mu dZ_\mu \quad \text{Ornstein-Uhlenbeck}$$

$$m_t = \frac{1}{\tau} \int_{t-\tau}^t \frac{dS_u}{S_u} \quad \text{Momentum}$$

and compute the optimal strategy, for a 2-asset universe (stock and bond) and an investor maximizing the expected utility of final wealth. The model can be fitted to historical data (S&P 500, monthly, 1 century).

Discounting the distant future
J.D. Farmer et al. (2014)

If the interest rate (is non-constant and) follows an Ornstein-Uhlenbeck process $dr = -\alpha(r - m)dt + k dW$, the discount factor $D(t) = E[\exp - \int_0^t r(s)ds]$ satisfies $\log D(t) \sim_{t \rightarrow \infty} (m - k^2/2\alpha^2)t$, in particular, the long-term interest rate $r_\infty = m - k^2/2\alpha^2$ is not the long-term average of the interest rate. The interest rate distribution, $r \sim N(m, k^2/2\alpha)$, does not determine r_∞ .

Independence of stock markets before and after the global financial crisis of 2007
B.M. Ibrahim and J. Brzezczński (2014)

To measure the impact of a variable z on a regression $y \sim x$, where x, y, z are time series, assume that the coefficients of the regression are AR(1) processes whose coefficients depend on Z ,

$$\begin{aligned} y_t &= (\alpha + \varepsilon_t) + (\beta + \eta_t)x_t + u_t \\ \varepsilon_{t+1} &= (a + bz_t)\varepsilon_t + v_{t+1} \\ \eta_{t+1} &= (c + dz_t)\eta_t + w_{t+1} \end{aligned}$$

For instance, x and y could be the returns in two markets and z the difference in turnover of volatility between those two markets.

An empirical investigation of methods to reduce transaction costs
T. Moorman (2014)

To reduce transaction costs, define a no-trade zone around the optimal portfolio, using some measure of distance (Euclidian, cosine, etc.) and an “optimal” threshold.

Causal dependency in extreme returns
K. Echaust

There is more long memory in return tails: look at the cross-correlation of block maxima and minima.

Option pricing with the logistic return distribution
M. Levy and H. Levy

Black Scholes option pricing formulas can be generalized to the logistic distribution (the distribution whose

cdf is the logistic function). A possible justification is that while aggregated log-returns are asymptotically Gaussian, normal market conditions are still far away from this limit.

Forecasting crashes: correlated fund flows and the skewness in stock returns
X. Gong et al. (2014)

Correlated demand for liquidity (from mutual funds) explains the negative skewness of stock returns:

- Use mutual fund holdings to estimate how much of the trading of a fund is due to liquidity shocks (inflows and outflows);
- Assume that the stock returns are linearly related to those liquidity shocks;
- Estimate the variance and skewness of the returns, conditional on the liquidity shocks;
- Decompose it into contributions of the variance, covariance, skewness, coskewness of the liquidity shocks.

Non-linear forecasting of energy futures: oil, coal and natural gas
G.G. Creamer (2014)

Use Brownian distance correlation

$$\begin{aligned} \nu(X, Y) &= \|f_X f_Y - f_{XY}\| \\ c(X, Y) &= \frac{\nu(X, Y)}{\sqrt{\nu(X)\nu(Y)}} \end{aligned}$$

as an alternative to Granger causality, and for feature selection (to build leading indicators).

Agent-based models of financial markets
E. Samanidou et al. (2007)

Clear review of a few agent-based models of financial markets:

- Rebalancers (targeting $w_{\text{cash}} = w_{\text{stock}} = \frac{1}{2}$) and CPPI investors, reviewing their portfolios (and trading) at random times, each agent estimating the “fair price” from the order book to make her decision, with random deposits and withdrawals;
- Traders with a noisy log-utility (so that the optimal weight of stock and bond is independent of wealth, up to noise), assuming the forward returns will be one of the past k returns (the look-back k is investor-dependent);
- Gamblers, $\text{wealth}_{i,t+1} = \lambda \times \text{wealth}_{i,t}$, with λ random around 1, and a welfare state ensuring $\text{wealth}_{i,t} \geq q \times \text{average wealth}$;
- *Percolation*: traders are the occupied sites on a lattice of dimension 2 to 7, clusters are groups of traders making the same decision (herding), $P[\text{buy}] = P[\text{sell}] = a$, $P[\text{sleep}] = 1 - 2a$, $\Delta \log \text{price} \propto \text{supply} - \text{demand}$; this gives the correct stylized facts, including log-periodic oscillations (if you add some rationality and hysteria);

- The supply and demand balance between noise traders and fundamentalists drives price changes, and each trader reviews her beliefs based on those price changes.

Only the last two models give credible prices.

Electricity price forecasting: a review of the state-of-the-art with a look into the future

R. Weron (2014)

Review of many methods to forecast electricity prices:

- Agent models (Nash equilibrium, multi-agent simulations);
- Jump diffusion model, Markov-switching model;
- Exponential smoothing, regression, AR, SARMA, ARX, TAR, GARCH, etc.;
- Neural nets (MLP, recurrent, fuzzy), SVM.

Financial sector tail risk and real economic activity: evidence from the option market

M. Neumann

An option-implied measure of tail risk can be computed from the price of deep OTM put options on a financial index (cheaper than it should be because of (implicit) government guarantee) and the constituents of this index.

Q-learning-based financial trading systems with applications

M. Corazza and F. Bertoluzzo (2014)

Use reinforcement learning (e.g., Q-learning) to design your automated trading systems.

TrueSkillTM: a Bayesian skill rating system

R. Herbrich et al.

Generalization of the (already Bayesian) Elo system.

Intrinsic ratings compendium

K.W. Regan (2012)

To estimate Elo ratings from moves rather than game outcomes (there are more of them):

- Compute the value of the position before and after each move (with a chess program);
- Assume $P[\text{move}] \propto p_0^{\alpha(\text{value}, \Delta\text{value})}$, where α is a function to estimate;
- Discard openings (first 8 moves), repeated positions, clearly advantageous positions;
- Estimate the Elo rating from α .

A different angle on fitting ROC curves to rating data using the first principal component

J.R. Vokey (2014)

To estimate a ROC curve from a few points, convert the p -values to a Gaussian (qnorm – the points are likely to be aligned) and take the first principal component (rather than a regression line).

From Archimedian to Liouville copulas

A.J. McNeil and J. Nešlehová

Archimedian copulas are copulas of d -dimensional ℓ_1 -norm symmetric distributions, *i.e.*, copulas of random variables $X = RS_d$, where S_d is uniform on the unit simplex $\Delta_d = \{x \in \mathbf{R}_+^d : \sum x_i = 1\}$ and R is an independent nonnegative random variable. Replacing the uniform distribution on Δ_d with a Dirichlet distribution, $P(x_1, \dots, x_n) \propto \prod x_i^{\alpha_i - 1}$ gives the (non-exchangeable) *Liouville copulas*.

An introduction to state space models

M. Wildi (2013)

Detailed presentation of state space models and the Kalman filter in R, with

- Detailed explanations of the formulas (the Kalman filter is a sequence of Bayesian updates – in particular, a prior is needed);
- The various objective functions for model estimation (look at the out-of-sample residuals $y_t = \hat{y}_{t|t-1}$, or the in-sample ones, $y_t = \hat{y}_{t|t}$, and minimize their sum of squares or their likelihood – prefer the out-of-sample maximum likelihood estimator);
- A few examples:
 - Regression, where the coefficients are the hidden state, and the covariates form the (time-changing) observation matrix;
 - Intervention studies;
 - Missing values, dealt with by skipping the observation $\hat{\xi}_{t|t} \leftarrow \hat{\xi}_{t|t-1}$;
 - Time-changing AR(1) process;
 - Decomposition into trend and SARMA components.
- Smoothing was not detailed – there are many applications in signal processing, but few/none in time series analysis.

In R, check the `dlm`, `KFAS`, `dlmodeler` packages.

Parallel Markov chain Monte Carlo

D.N. VanDerwerken and S.C. Schmidler (2013)

Use importance sampling weights to merge parallel MCMC simulations and get a consistent estimator much earlier, even in case of slow mixing (e.g., multimodal or highly correlated distributions).

Testing the equality of two positive definite matrices with applications to information matrix testing

J.S. Cho and H. White (2014)

Build a test for equality of positive definite matrices using

$$A = B \iff (\det D)^{1/k} = \frac{1}{k} \text{tr } D = 1,$$

where $D = BA^{-1}$.

***Birds of a feather flock together
Local learning of mid-level representations
for fine-grained recognition***
A. Freytag et al.

Local learning is similar to local regression, but for arbitrary machine learning algorithms: given a new observation to classify, find the k -nearest ones ($k \gg 1$), train a model on them, and use it.

***Multivariate Weibull distributions
for asset returns I***
Y. Malevergne and D. Sornette (2004)

Asset returns X can be described by a modified Weibull distribution,

$$p(x) \propto |x|^{c/2-1} \exp - \left(\frac{|x|}{\sigma} \right)^c$$

(perhaps with a different exponent c for $x > 0$ and $x < 0$), *i.e.*, some power of the returns, $\text{sign}(X)|X|^\alpha$, is Gaussian. For the correlation structure, use a Gaussian copula.

***Tweedie family densities:
methods of evaluation***
P.K. Dunn and G.K. Smyth

The Tweedie distribution is an exponential dispersion model in which the variance is some power of the expectation

$$f(y) = a(y, \phi) \exp \frac{y\theta - \kappa(\theta)}{\phi}$$

$$\text{Var } y \propto E[Y]^p.$$

Special cases include Gaussian ($p = 0$), Poisson ($p = 1$), Gamma ($p = 2$) and inverse Gaussian ($p = 3$). The pdf can be approximated by inverting the cumulant generating function (it has a simple form) or by an infinite series.

In R, check the `cplm` and `tweedie` packages.

***Series evaluation of Tweedie exponential
dispersion model densities***
P.K. Dunn and G.K. Smyth (2005)

If $Y \sim \text{ED}_p(\mu, \phi)$ follows a Tweedie distribution with $0 < p < 1$, it can be written $Y = X_1 + \dots + X_N$, with $X_i \sim \Gamma$, X_i iid, $N \sim \text{Poisson}$. When p is close to 1, the distribution is multimodal ($p = 1$ gives the Poisson distribution, which is discrete).

***Model identification
using stochastic differential equation
grey-box models in diabetes***
A.K. Duun-Henriksen et al. (2013)

A white-box model is a system of ODEs (here, pharmacokinetics) based on domain knowledge. A grey-box model is a noisy system of SDEs; it mixes

domain knowledge and data. A black-box model is entirely data-driven.

For grey-box models in R, check the `ctsm` package (SDEs with noisy measurements at discrete times, *i.e.*, non-linear Kalman filter).

***Approximate bayesian inference
for latent Gaussian models by using
integrated nested Laplace approximations***
H. Rue et al. (2009)

One can use *nested* Laplace approximations (asymptotic expansion of $\int_a^b e^{Mf(x)} dx$, when $M \rightarrow \infty$, if f has a unique maximum x_0 , using a Taylor expansion of f around x_0) to estimate posterior probabilities in latent Gaussian models.

In R, check the `INLA` package (not on CRAN).

***Non-linear causal inference
using Gaussianity measures***
D. Hernández-Lobato et al. (2014)

If x causes y (through a linear model with non-Gaussian noise), the residuals of $y \sim x$ are less Gaussian than those of $x \sim y$.

***Ranking the economic importance
of countries and industries***
W. Li et al. (2014)

Country- and sector-level imports and exports (world input-output data, www.wiod.org) defines a (weighted, directed) graph, amenable to *cascading failure tolerance analysis*: check what happens when a node fails – other nodes fail if their revenue drops by a fraction greater than $p \in [0, 1]$. The critical value p_c beyond which most nodes fail measures the centrality of the first failing node.

***Interdependencies and causalities
in coupled financial networks***
I. Vodenska et al. (2014)

To identify lead-lag relations from log-return time series, complexify the signals with a Hilbert transform; compute the “complex correlation matrix”; build a directed network using the sign of the phase of the correlation to infer the direction of the edges. One can also look at the principal components (complex PCA) and use random matrix theory (or its resampling-based equivalent, rotational random shuffling, RRS).

***SIMoNe: statistical inference
for modular networks***
J. Chiquet et al. (2008)

If you suspect your network has a modular structure, *i.e.*, the adjacency matrix has a block structure, with dense diagonal blocks and sparse off-diagonal blocks.

***Maps of random walks on complex networks
reveal community structure***

M. Roswall and C.T. Bergstrom (2008)

One can describe paths on a graph by using a Huffman code for the outgoing edges of each node – but there is a different code for each node. Instead, one can use a unique code, and encode the target nodes instead of the outgoing edges (Huffman code for the limiting distribution of a random walk on the graph). To identify clusters, use a 2-level code: each cluster has a unique name, each node has a code unique in its cluster, but reused on other clusters. The path is encoded by using the node codes, as long as it remains in the same cluster and, when it changes cluster, an exit code, the name of the cluster, and the code of the new node. The average description length is

$$P[\text{change cluster}]H[\text{cluster codes}] + \sum_{\text{cluster}} P[\text{same cluster}]H[\text{node codes in that cluster}]$$

where H is the entropy.

To find the optimal partition into clusters, use a greedy agglomerative algorithm, and refine with simulated annealing. Contrary to other clustering algorithms, the edge weights are used. By adding teleportation (à la PageRank), the algorithm can also deal with directed graphs.

***Online community detection
for large complex networks***
G. Pan et al. (2014)

To estimate communities online, do not optimize the modularity, but the expected modularity, for some generative model, e.g.:

- Link a new node to a community C with probability proportional to $\deg C$;
- Link two nodes in communities C and C' with probability proportional to $\deg C \times \deg C'$.

***Nowcasting economic and social data:
when and why search engine data fails,
an illustration using Google flu trends***
P. Ormerod et al. (2014)

Search engine data (e.g., Google flu) is less reliable when social influence is high: the *Bass diffusion model* (fitted between the two low points on both sides of a peak) can help distinguish between independently-motivated and socially-motivated searches.

***Do we need hundreds of classifiers
to solve real-world classification problems?***
M. Fernández-Delgado et al. (2014)

Comparison of 179 classifiers (the output is binary), in R, Weka, C, Matlab (no Python) on 121 datasets: prefer random forests and SVM, but do not reject neural networks, boosting, C5.0, avNNet (caret) or ELM (extreme learning machines).

***Generating abbreviations
using Google Books library***

V.D. Solovyev and V.V. Bochkarev

Abbreviations are prefixes (or substrings) with the same context.

***Improved part-of-speech tagging for online
conversational text with word clusters***
O. Owoputi et al. (2013)

TweetNLP is a POS tagger for twitterese; it uses a hidden Markov model (HMM) with Brown clustering to recognize new words and alternate spellings.

***Selecting influential examples: active learning
with expected model output changes***
A. Freytag et al.

Active learning refers to the algorithms used to decide which unlabeled samples to label next (when labeling is costly and online), *i.e.*, which sample is the most informative. Strategies include:

- Rapid exploration: prefer samples far away from already-labeled ones – but this gives too much emphasis on outliers;
- Maximum uncertainty: label the samples the model is the most uncertain about;
- Maximize the expected model change;
- Reduce the estimated classification error (expected entropy minimization);
- Maximize the expected model output change (EMOC).

Logarithmic-time online multiclass prediction
A. Choromanska and J. Langford (2014)

For multiclass classification problems with a large number of classes, arrange the classes in a logarithmic-depth tree, either:

- Known in advance;
- Constructed using class frequencies (Huffman coding);
- Constructed online, recycling orphan nodes when needed.

Model compression
C. Bucilă et al. (2006)

To compress an ensemble of thousands of models into a simpler, smaller model, fit the smaller model on the forecasts of the complicated model on a large set of unlabeled examples. If there is no such set, use a synthetic one: add noise to each feature of existing observations, where the amplitude of the noise is the distance to the closest observation (for qualitative variables, replace the value with that of the nearest neighbour with probability p).

Do deep nets really need to be deep?

L.J. Ba and R. Caruana (2014)

Shallow neural networks are as expressive as deep ones:

- Learn a deep model;
- Use it to create a huge synthetic training set from unlabeled data;
- Train a shallow model on the synthetic dataset (to speed things up, use a low-rank approximation of the weight matrix).

Towards automated discovery of artistic influence

B. Saleh et al. (2014)

Linguistic models can be used with non-text data: for instance, one can use topic models (latent Dirichlet analysis, LDA) on images, by replacing the words with image features (first identified by the Laplace Harris detector, then reduced to a set of 600 features (“words”) using k -means).

Memory networks

J. Weston et al. (2014)

One can combine neural networks with a memory component:

- Convert the input to features;
- Store them in memory (more generally, update the memory from the input features and the current memory);
- Predict the output features from the memory and the input text;
- Generate the output from the output features.

The various components can use your favourite machine learning algorithm (SVN, decision tree, neural net, etc.). The idea can be used on text data, using a bag-of-words approach, e.g., to answer questions about a story (store the raw input, verbatim, sequentially, time-stamped, in memory; compute the memory m_1 “closest” to the input question; compute the memory m_2 closest to both the input and m_1 and with $m_1 \prec m_2$; output a single word).


Modeling interestingness with deep neural networks

J. Gao et al. (2014)

Use a deep neural net, with a convolutional and a max-pooling layer (often used in image processing) to model text; train it to forecast followed links, *i.e.*, $f(\text{text}_1, \text{text}_2) = 1$ if text_2 was read after text_1 .

Understanding locally competitive networks

R.K. Srivastava et al.

Visualizing the nodes of a neural network with rectified linear units (RLU, ) , maxout (output the maximum of the inputs) or local winner takes all (in a group of nodes, the maximum passes its input unchanged, but the others output 0) (these are often trained with

droupout) suggest that the nodes cluster – only part of the network is activated for any given input pattern.

Political ideology detection using recursive neural networks

M. Iyyer et al.

A **recursive neural net** (RNN) is a model that

- Uses a vector representation of the words in the sentence (initialize it with word2vec);
- Combines them, along the parse tree of the sentence (all nodes share the same parameters);
- And outputs the predicted class (here, liberal vs conservative) using a softmax.

Efficient programmable learning to search

H. Daumé III et al. (2014)

In structured prediction, the search space can be described by an arbitrary imperative program (or a grammar).

Software to model sequence data (segmentation, tagging in NLP): CRF++, crfsgd, VW.

Dropout: a simple way to prevent neural networks from overfitting

N. Srivastava et al. (2014)

When iteratively training a neural net (e.g., with stochastic gradient descent, SGD), forget (drop) a random subset of the nodes at each iteration (and update this smaller set). The resulting neural net can be seen as an ensemble of (all the) 2^n sparser subnets.

This can be interpreted as the addition of noise to the hidden nodes; this noise can be integrated out to compute the gradient (for regression, this is similar to ridge regression).

Monte Carlo non-local means: random sampling for large-scale image filtering

S.H. Chan et al. (2013)

To denoise a pixel i in an image, take the weighted average of the pixels j , $\sum_j w_{ij} x_j$, where w_{ij} measures the similarity of the regions around i and j (non-local means, NLM); the pixels j can be from the same image or from an image database. To speed up the algorithm, consider:

- Only looking at pixels j close to i ;
- Dimension reduction (SVD);
- Effective data structures (kd-trees);
- A low rank approximation of the weight matrix W ;
- Random sampling.

Modeling the shape of the scene: a holistic representation of the spatial envelope

A. Oliva and A. Torralba (2000)

It is possible to recognize scenes by looking at their texture and structure, without any segmentation (GIST descriptors):

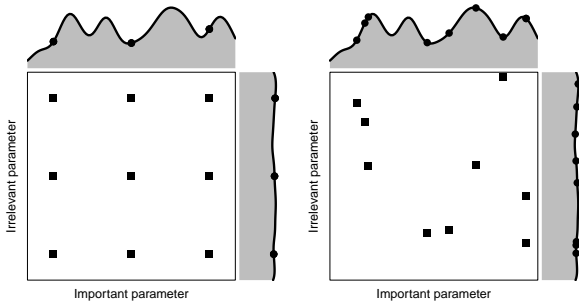
- Presence of textured zones;
- Undulating contours;
- Uncluttered horizon line;
- Size of the major components;
- How straight the horizon line is;
- Parallels and perpendiculars;
- Converging lines.

***Evaluation of GIST descriptors
for web-scale image search***
M. Douze et al. (2009)

Useful for duplicated image search.

***Random search
for hyper-parameter optimization***
J. Bergstra and Y. Bengio (2012)

Avoid grid search for hyper-parameter optimization: if some of the directions are irrelevant, random search (or a low discrepancy sequence) performs better.



Freeze-thaw bayesian optimization
K. Swersky et al. (2014)

Bayesian optimization searches the (global) minimum of an expensive, noisy function (over $[0, 1]^D$). The function is often modeled as a Gaussian process (GP). To choose the next point on which to evaluate the function, one can use:

- The point that gives the best expected improvement

$$a_{EI}(x) = (f(x_{\text{best}}) - \mu(x))_+$$

(where the positive part is smoothed – they do not use $u_+ \approx \log(1 + e^u)$, but $u_+ \approx u\Phi(u) + \phi(u)$ – and the smoothing is larger when the variance $V(x)$ is large);

- The point that reduces the expected uncertainty about the location of the minimum the most (entropy search).

It is actually not needed to precisely estimate the function at each point: one can stop early (freeze), estimate the limiting value (by assuming exponential convergence – this is not unlike the classical series acceleration techniques) and, at each iteration, decide to either explore a new point, or continue (thaw) the computations for a frozen point.

For bayesian optimization in Python, check HIPS/spearmint.

***MA-SW-Chains: memetic algorithm
based on local search chains
for large scale continuous global optimization***
D. Molina et al. (2010)

A memetic algorithm is an evolutionary algorithm, hybridized with local search. The chains refer to the fact that the local search is not run until the end, but just a few more steps at each generation.

***Approximate bayesian computation
and particle filters***
D. Prangle (2014)

Maximum likelihood estimation can still be used when the likelihood is not computable:

- Simulate data y_{sim} for many values of the parameter θ ;
- Select the θ for which y_{sim} and y_{obs} are the closest (instead of using all of y , you can use some summary statistic).

In a bayesian framework, this likelihood-free method becomes “approximate bayesian computation” (ABC):

Repeat N times:

- Draw θ from the prior
- Draw $y_{\text{sim}}|\theta$ from the model
- Accept θ if $d(y_{\text{sim}}, y_{\text{obs}}) < \varepsilon$.

To avoid degeneracy (no samples accepted), repeat until there are M samples (alive ABC).

Adding noise makes the estimate consistent (noisy ABC).

For state space models, one can start with a *particle filter*

$t = 1$

Sample $x^{(i)}$ from the prior

$w^{(i)} \propto \text{likelihood}(y_t^{\text{obs}}|x^{(i)})$

Increment t , resample, propagate the particles

and replace the likelihood

$$\begin{aligned} &\text{Simulate } y^{(i)}|x^{(i)}, \theta \\ &w^{(i)} = \mathbf{1}_{d(y_t^{\text{obs}}, y^{(i)}) < \varepsilon} \end{aligned}$$

***Accurate methods
for approximate bayesian computation filtering***
L. Calvet and V. Czellar (2012)

In the ABC particle filter

$$p_t^{(n)} \propto \mathbf{1}_{d(\tilde{y}_t^{(n)}, y_t) < \varepsilon},$$

where y is the data and \tilde{y} is the data sampled from the model (N samples), choosing ε is akin to selecting the bandwidth of a kernel density estimator (this guarantees $\varepsilon \xrightarrow[N \rightarrow \infty]{} 0$).

Relevant statistics for Bayesian model choice
J.M. Marin et al. (2013)

The choice of a (non-sufficient) summary statistic T (i.e., using $\pi(\theta|T(y))$ instead of $\pi(\theta|y)$, where θ =model, y =data, T =summary statistic) in bayesian statistics (in particular ABC) leads to incorrect Bayes factors, and therefore incorrect model selection. Under some conditions on the summary statistics, the Bayes factor is asymptotically correct.

Bayesian computation via empirical likelihood
K.L. Mengersen et al. (2012)

The *empirical likelihood* (in R, `empLik`) is an alternative to approximate bayesian computations (ABC): given a set of equations uniquely determining the parameters of interest θ , $E[h(y, \theta)] = 0$, let

$$L_{\text{el}} = \text{Max} \left\{ \prod_i p_i : p \in [0, 1]^n, \sum p_i = 1, \sum p_i h(y_i, \theta) = 0 \right\}.$$

Multidimensional scaling using majorization: SMACOF in R
J. de Leeuw and P. Mair

To find the minimum of a function f , *majorization* suggests to find a simpler function g such that

$$\begin{aligned} \forall x, y \quad g(x, y) &\geq f(x) \\ \forall y \quad g(y, y) &= f(y) \end{aligned}$$

and iterate $x_{n+1} = \text{Argmin}_x g(x, x_n)$.

Multi-dimensional scaling (MDS) is the problem

$$\text{Argmin}_{x_1, \dots, x_n} \sum_{ij} w_{ij} (\delta_{ij} - \|x_i - x_j\|)^2;$$

the objective function can be written

$$\begin{aligned} \sigma(X) &= 1 + \text{tr } X' V X - 2 \text{tr } X' B(X) X \\ &\geq 1 + \text{tr } X' V X - 2 \text{tr } X' B(X) Y \end{aligned}$$

(the rhs is now a quadratic function).

Variants include:

- Several dissimilarity matrices;
- Linear constraints (they can be formulated as a reparametrization: find C to minimize $\sigma(ZC)$, with Z known);
- Rectangular matrices: given a judge×object rating matrix, find coordinates for both judges and items in the same space – this is a special case of the previous setup, with weights $W = \begin{pmatrix} 0 & * \\ * & 0 \end{pmatrix}$;
- Non-metric MDS:

$$\text{Argmin}_{\substack{X, f \\ f \text{ non-decreasing}}} \sum_{ij} w_{ij} (f(\delta_{ij}) - \|x_i - x_j\|)^2,$$

the estimate of f changes at each step, and is obtained by *monotone regression* (aka *isotonic regression*)

- MDS on quadratic surfaces (mostly spheres).

Hierarchical k-means clustering
A. Böcker et al. (2004)

The hierarchical k -means algorithm is a sequence of k -means invocations:

- Start with the mean;
- Split all the centroids in two, add noise, and start the k -means algorithm with these centroids;
- Iterate until the number of centroids equals the number of points.

Clustering with Bregman divergences
A. Banerjee et al. (2005)

The k -means algorithm (of the k -medoid one – the cluster centers are chosen among the data points) can be generalized to other distances or divergences. The *Bregman divergence* associated to a convex function $\phi : S \rightarrow \mathbf{R}$, with $S \subset \mathbf{R}^n$, is

$$d_\phi(x, y) = \phi(x) - \phi(y) - \langle x - y, \nabla \phi(y) \rangle.$$

Examples include $\phi(x) = \|x\|^2$, for the Euclidian distance; $\phi(x) = \sum x_i \log x_i$, for the Kullback-Leibler divergence on the simplex $S = [\sum x_i = 1, \forall i \ x_i \geq 0]$; $\phi(x) = -\log x$, for the Itakura-Saito distance,

$$D(p, q) = \sum_i \left(\frac{p_i}{q_i} - \log \frac{p_i}{q_i} - 1 \right).$$

Itakura-Saito nonnegative matrix factorization and friends for music signal decomposition
C. Févotte (2012)

The β -divergence,

$$d_\beta(x|y) = \frac{1}{\beta(\beta-1)} (x^\beta + (\beta-1)y^\beta - \beta xy^{\beta-1}),$$

which specialized to the Euclidian distance ($\beta = 2$), the Kullback-Leibler divergence ($\beta = 1$) or the Itakura-Saito divergence ($\beta = 0$, non-convex, but scale-invariant) is a popular cost function in nonnegative matrix factorization (NMF). It is non-convex in general, but can be optimized by majorization-minimization (MM) by majorizing the convex and concave parts separately (using Jensen's inequality and the tangent, respectively). IS divergence NMF is actually a maximum likelihood estimator for a generative model of the short-term Fourier transform (STFT) of the signal.

In music, NMF can recover instruments and notes.

***On the surprising behavior of distance metrics
in high-dimensional space***
C.C. Aggarwal et al. (2009)

The curse of dimensionality states that, in high dimensions, the closest point and the farthest point are approximately at the same distance. A study of the asymptotic behaviour of

$$\frac{\max_{1 \leq i \leq n} \|X_i\|_{L^k}}{\min_{1 \leq i \leq n} \|X_i\|_{L^k}}$$

as the dimension d tends to infinity suggests that the situation is worse if k is large. Prefer the L^1 distance or, even better, the L^k semi-metric for $k \in (0, 1)$. It is not a metric – it does not satisfy the triangle inequality – but, for algorithmic purposes (e.g., to look for the k nearest neighbours), it is usually fine.

***Feature selection strategies for classifying
high-dimensional astronomical data sets***
C. Donalek et al.

Algorithms to select features include:

- Fast relief: compare the distribution of $f(\text{nearest hit}) - f(x_0)$ and $f(\text{nearest miss}) - f(x_0)$;
- Fisher discriminant ratio: $\text{FDR} = \left(\frac{\mu_1 - \mu_2}{\sigma_1/\sigma_2} \right)^2$;
- Correlation-based feature selection: find a subset of the variables with low correlation between themselves and high correlation with the target;
- Least angle regression.

***Distribution's template estimate
with Wasserstein metrics***
E. Boissard et al. (2013)

The notion of mean (and barycenter) can be generalized to arbitrary metric spaces,

$$(x_1, \dots, x_n) = \underset{\mu}{\text{Argmin}} \sum_i d(\mu, x_i)^2,$$

but unicity, or even existence, is not guaranteed in general. Given a metric measurable space (E, d, Ω) , the Wasserstein distance between two probability measures μ and ν on E is

$$d(\mu, \nu)^2 = \inf_{\substack{m \text{ measure on } E \times E \\ \text{pr}_1 m = \mu \\ \text{pr}_2 m = \nu}} \int_{E \times E} d(x, y) m(dx, dy).$$

If $E = \mathbf{R}^n$ and μ is absolutely continuous wrt the Lebesgue measure, there exists $T : \mathbf{R}^n \rightarrow \mathbf{R}^n$ (Brenier map) such that $\nu = T * \mu$ and

$$d(\mu, \nu) = \int |T(x) - x|^2 \mu(dx).$$

It measures the deformation needed to transform μ into ν and can be used to compute barycenters.

***Item response models
to measure corporate social responsibility***
N. marco et al. (2013)

To combine several signals into one (here, various CSR measures, provided by KLD – now MSCI ESG), one can use:

- Equal weights;
- Weights reflecting stakeholders' preferences;
- Data envelopment analysis (DEA – the weights are company-specific and determined by the data);
- The first principal component, from PCA, ICA, NMF, etc. [not mentioned];
- Item response theory (IRT): this automatically identifies the offset for each variable and how discriminative they are.

***Measuring corporate social performance:
an efficiency perspective***
C.M. Chen and M. Delmas (2010)

Data envelopment analysis (DEA) is a way of measuring the distance (of a firm) from the efficient frontier (of its peers). Let i denote companies, k the measures to be maximized (profit, etc.), x_{ki} the value of measure k for firm i , $F = \{i : \forall j \exists k x_{ki} \geq x_{kj}\}$ the efficient frontier. The aggregated score for company i is $\sum_k \lambda_{ki} x_{ki}$, where the optimal weights are

$$\lambda_{\cdot i} = \underset{\substack{\lambda_{\cdot i} \\ \lambda \geq 0 \\ \sum_k \lambda_{ki} = 1}}{\text{Argmin}} \max_{j \in F} \sum_k \lambda_{ki} (x_{kj} - x_{xi}).$$

Traditionnaly, DEA separates the measures into inputs x_{ki} (to be minimized) and outputs x_{kj} (to be maximized):

$$\begin{aligned} \text{Find} & \quad \lambda_{\cdot i}, \mu_{\cdot i} \\ \text{To maximize} & \quad \sum_{\ell} \mu_{\ell i} y_{\ell i} / \sum_k \lambda_{ki} x_{ki} \\ \text{Such that} & \quad \forall j \quad \sum_{\ell} \mu_{\ell i} y_{\ell j} / \sum_k \lambda_{ki} x_{kj} \leq 1 \\ \text{and} & \quad \lambda, \mu \geq 0. \end{aligned}$$

(To make the program linear, rescale it by adding the constraint $\sum_k \lambda_{ki} x_{xi} = 1$.)

The ratio $\sum_{\ell} \mu_{\ell i} y_{\ell i} / \sum_k \lambda_{ki} x_{ki}$ is the efficiency of the firm; it is 1 iff it is on the efficient frontier.

The article uses an extension of DEA to ordinal variables to aggregate corporate social responsibility measures.

[Would the bootstrap help give more stable results?]

***Non-linear shrinkage of the covariance matrix
for portfolio selection:
Markowitz meets Goldilocks***
O. Ledoit and M. Wolf (2014)

Linear shrinkage estimates a covariance matrix as $(1 - \lambda)V + \lambda I$, where V is the sample variance matrix

and λ is chosen to minimize some loss function. The loss function could be the asymptotic Frobenius distance between the estimator and the population variance (the population variance is not known but, to compute the *asymptotic* distribution, we do not need all of it: the eigenvalues actually suffice). In a financial context, one can minimize the variance of the tangential portfolio: the results turn out to be similar.

Instead of letting just one parameter vary, one can let $O(N)$ parameters vary, e.g., the eigenvalues of the sample covariance matrix.

The estimator is defined as follows:

$c = N/T$ concentration
 $V = U\Lambda U'$ sample covariance matrix
 $\lambda_1, \dots, \lambda_n$ eigenvalues of V
 τ_1, \dots, τ_n eigenvalues of the population variance

$s(x) \in \mathbf{R} \cup \mathbf{C}^+$ solution of $s^{-1} = \frac{1}{T} \sum_{i=1}^N \frac{\tau_i}{1 + \tau_i s} - x$

$d_i = \frac{1}{\lambda_i s(\lambda_i)}$ new eigenvalues

$D = \text{diag}(d_1, \dots, d_n)$

$S = UDU'$ desired estimator.

The population eigenvalues can be estimated by discretizing the relation between the asymptotic empirical distribution function H of the population eigenvalues and that, F , of the sample eigenvalues: for all $x \in \mathbf{C}^+$, the Stieltjes transform of F ,

$$s = \int \frac{dF(\lambda)}{\lambda - z}$$

is determined by

$$-\frac{1-c}{z} + cs \in \mathbf{C}^+ \\ s = \int \frac{dH(\tau)}{\tau(1-c-czs) - z}.$$

Linear vs quadratic portfolio selection models in practice F. Cesarone et al.

The standard quadratic programming problem

Find $x \in \mathbf{R}^n$
 To minimize $f(x) = x'Qx$
 Such that $x \geq 0, x'1 = 1$

can be solved efficiently with the *increasing set* algorithm. The minimum is attained in the relative interior of a face where f is strictly convex; furthermore, there is an increasing sequence of faces $F_1 \subset F_2 \subset \dots \subset F_k$, $\dim F_i = i$, on which f is strictly convex, has an interior global minimizer x_i^* and $x^* = x_k^*$. The algorithm looks at all possible faces of dimension 1 satisfying the conditions, and tries to extend them (breadth-first).

For cardinality-constrained problems, that is even easier: stop early.

This can be used to find cardinality-constrained minimum variance portfolios.

Premium indexation: CVaR-lasso optimal index replication Y.J. Jin and X. Zhong (2014)

To replicate an index, use CVaR optimization rather than mean-variance and add an L^1 constraint (the authors suggest a hard constraint, $\sum_i |w_i| \leq 2$) to control the number of stocks in the replicating portfolio.

Portfolio selection under directional predictability of returns J. Hämäläinen (2014)

Forecasting return signs is easier than forecasting returns, and the signs can still be used in a mean-variance optimization: the retrun distributions can be computed using

- Sign forecasts, s_i for sign X_i ;
- The probability that they are correct, $p_i = P[\text{sign } X_i = s_i]$;
- The probability that they are jointly correct or incorrect, $p_{ij} = P[(\text{sign } X_i)(\text{sign } X_j) = s_i s_j]$.

Diversification management of a multi-asset portfolio M. Poonia (2014)

Comparison of various notions of diversification: number of stocks, diversification ratio, risk parity, PCA risk diversification, torsion bets.

Conditional Sharpe ratios K.V. Chow and W.C. Lai

When computing the Sharpe ratio, replace the excess returns by 0 unless the benchmark returns are below their α quantile.

Portfolio choice in the presence of estimation error: a pricing model filter approach M. Lozano (2014)

Use the sample covariance matrix of the CAPM (or some factor model) residuals.

Sparse and robust normal and t-portfolios by penalized Lq likelihood minimization D. Ferrari et al. (2014)

The lasso can be generalized to

$$\hat{\beta} = \underset{\beta, \mu, \sigma}{\text{Argmin}} - \sum_{i=1}^n L_q f \frac{x_i' \beta - \mu}{\sigma} - \sum_{j=1}^p L_p \pi(\beta_j, \lambda)$$

where

x_i = observations

f = Gaussian distribution (for instance)

L_q = generalized logarithm,

$$L_q(z) = \begin{cases} \frac{z^{q-1} - 1}{1 - q} & \text{if } q \neq 1 \\ \log z & \text{if } q = 1 \end{cases}$$

π = prior on the parameters, e.g., Laplace,

$$\pi(\beta, \lambda) = \lambda e^{-\lambda|\beta|/2}.$$

The lasso is obtained for $q = 1$, π = Laplace, f = Gaussian.

***Portfolio spillovers
and a limit to diversification***
B. Argyle (2013)

After a sharp price change to a stock, investors holding the stock will rebalance their portfolio, but not necessarily by buying/selling this stock – it may be easier, faster to buy/sell more liquid assets. In the case of large institutional investors, and/or many investors holding a similar portfolio, there could be a price impact to those other stocks: can we detect it?

***Dynamic portfolio selection
by augmenting the asset space***
M.W. Brandt and P. Santa-Clara (2006)

To compute an (approximately) optimal dynamic asset allocation when all you can do is compute static asset allocations (Markowitz portfolio optimization), just add a few managed portfolios (e.g., weights proportional to some factor) to your universe of assets, and compute the corresponding “static” optimal portfolio. Use a VAR model if you end up with too many assets.

Tilt Nickel to Diamond
G. Xiang and T. Yu (2014)

Take a set of risk factors (Fama-French, Carhart, etc.) and a set of alpha factors. Starting with a capital-weighted portfolio (or some other benchmark), increase the exposure to the alpha factors while keeping the exposure to the risk factors close to zero.

***The deflated Sharpe ratio: correcting
for selection bias and backtest overfitting***
M. López de Prado (2014)

When computing the Sharpe ratio of the best strategy in your backtests, you can account for multiple testing and non-Gaussian returns as follows:

- Estimate the return distribution,
 $X_{it} \sim F(\mu, \sigma, s, \kappa)$ iid;
- Estimate the corresponding distribution of the (sample) Sharpe ratios S_i ;
- Estimate the distribution of the best one, $\max_{1 \leq i \leq n} S_i$;

- Convert the best Sharpe ratio into a p -value, using this distribution;
- Convert the p -value into a z -score – a deflated Sharpe ratio.

The independence assumption is unrealistic, in this context: when backtesting many strategies, they are often just variants of the same idea, so their returns are very similar.

Evaluating trading strategies
C.R. Harvey and Y. Liu (2014)

To compare several strategies, take multiple testing into account:

- FWER (family-wise error rate, e.g., Bonferroni or Holm correction) is often too conservative;
- FDR (false discovery rate, e.g., BHY correction) gives more usable results.

The discussion is needlessly complicated by the use of T values instead of p -values.

In R check `bonferroni`, `holm`, `BY`, in the `mutoss` package.

Optimal asset pricing
R. Turner et al. (2014)

Asset selling (sometimes called “asset pricing”) studies the optimal price of a perishable asset (airline ticket, hotel room, TV advertising slots) for which customers arrive randomly (inhomogeneous Poisson process) with (known) decreasing price elasticity (they are less and less price-sensitive as time passes).

Pricing policy $x_q(t)$ (price of the asset at time t if there are still q in inventory) and expected revenue $v_q(t)$ are related by a system of ODEs. For the optimal policy, just add

$$\frac{\partial R_q(x, t)}{\partial x} = 0,$$

where $R_q(x, t)$ is the expected revenue if a customer arrives at time t and is quoted price x .

Disentangling rebalancing return
W.G. Hallerbach (2014)

The “rebalancing return”, *i.e.*, the difference between a rebalanced portfolio and a buy-and-hold portfolio, can be decomposed into a volatility return (“volatility harvesting”) and a “dispersion discount” (a negative contribution), coming from the dispersion between the growth rate $\mu_i - \frac{1}{2}\sigma_i$ of the assets.

***Signal-wise performance attribution
for constrained portfolio optimization***
B. Durin (2014)

One can decompose the performance of a constrained portfolio built from several signals into the contribution of each signal and that of the constraints (with no residual term).

***Learning bayesian networks in R:
an example in systems biology***
M. Scutari (UseR 2013)

The **bnlearn** package can learn the structure of a Gaussian or discrete bayesian network, using

- Constraint-based algorithms (incremental association), that perform tests of conditional independence (mutual information, χ^2 , partial correlation);
- Score-based algorithms, that maximize (hill-climbing, tabu search) the goodness of fit (likelihood, AIC, BIC, Dirichlet posterior density, etc.) of the network;
- Hybrid algorithms, local search using statistical tests to define the neighbourhoods.

If the data is continuous but non-Gaussian, one can discretize it with *Hartemink's method*: discretize each variable into 60 quantile bins, collapse them while reducing the mutual information as little as possible, stop when each variable has 3 levels (low, average, high).

The **deal**, **catnet**, **pcalg** packages can also learn the graph structure, and some even accept mixed data (but each only provides one algorithm). All those packages can also fit the model parameters. For inference, use **gRain** (fast) or **bnlearn** (flexible).

Bootstrap can improve the stability of the result.

To model *interventions*, either whitelist all the edges from the intervention node, or blacklist all the edges to the intervention node.

```
b <- boot.strength(d, R=200, algorithm="hc")
g <- averaged.network(b, threshold=.85)
plot(cpdag(g)) # Remove ambiguous directions
```

***Learning bayesian networks
with the bnlearn R package***
M. Scutari (JSS, 2010)

More (but terse) details about the underlying algorithms, tests and scores.

***Bayesian networks in R
with applications in systems biology***
R. Nagarajan et al. (2013).

After a chapter on **bnlearn** (see the presentation above for a more thorough exposition), the book reviews, the book quickly presents dynamic networks. VAR models can be represented as dynamic bayesian networks (DBN), and estimated using shrinkage (**GeneNet**), lasso (**var**, **lars**, **glmnet**, **penalized**), low order independencies (**G1DBN**), change point models (**ARTIVA**), modular assumptions (**simone**).

The inference chapter names the main algorithms (variable elimination, junction trees, particle methods), with few details, and gives examples with **gRain** (**setFinding**, **querygrain**) and **bnlearn** (**cpdist**, **cpquery**).

***Inferring dynamic genetic networks
with low order independencies***
S. Lèbre (2009)

The structure of undirected Bayesian networks (random Markov fields) can be estimated from the partial correlation (which can be computed from the inverse of the correlation matrix, or from regressions): add an edge between i and j if $\text{pCor}(X_i, X_j) \neq 0$.

When there are not enough observations ($p \gg n$), one can use a shrinkage estimator (e.g., replace the correlation matrix C with $(1 - \lambda)C + \lambda I$ to make it invertible, or use a Lasso regression), or the first order partial correlation: add an edge between i and j if $\text{Cor}(X_i, X_j) \neq 0$ and $\forall k \neq i, j, \text{Cor}(X_i, X_j | X_k) \neq 0$ (the 0th order partial correlation is the correlation, the $(q+1)$ st order partial independence graph is a subgraph of the q th).

This can be used to infer the structure of *dynamic bayesian networks* (DBN). There is an implementation in the **G1DBN** R package.


***Inferring causal impact
using bayesian structural time-series models***
K.H. Brodersen et al. (2014)

To measure the effect of an intervention on a time series y_t , the difference-in-difference approach just compares

$$y_{\text{after}} - y_{\text{before}} \mid \text{intervention}$$

with

$$y_{\text{after}} - y_{\text{before}} \mid \text{no intervention,}$$

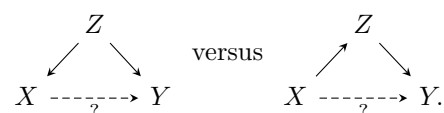
ignoring time series dynamics. Instead, one can model the time series, before the intervention, with a state space model, including a linear trend (whose slope is a random walk or a mean-reverting time series), seasonality, and covariates with dynamic coefficients. A *spike-and-slab* prior  provides automated variable selection. This is implemented in the **CausalImpact** R package, and used for A/B testing.

***Credit risk spillovers among financial
institutions around the global credit crisis:
firm-level evidence***
J. Yang and Y. Zhou (2013)

To study spillover risk, fit a VAR model to the data, and perform a “DAG analysis” on (the correlation matrix of) the residuals, *i.e.*, fit a bayesian network, and check if the direction of the edges is well-defined.

Reducing bias through directed acyclic graphs
I. Shrier and R.W. Platt (2008)

Identifying causal relations in observational data is tricky. Introducing a confounding variable in the regression can reduce or increase the bias, e.g.,



Real world causal networks are more complicated. If the DAG is known, one can find whether the variable will increase or decrease the bias as follows: the co-variables should not be descendants of X ; keep only ancestors of X , Y or Z ; delete arcs leaving X ; moralize, convert to undirected; delete Z ; if X and Y are no longer connected, then Z can be used as a confounder in $Y \sim X$.

There are potential problems: the DAG need not be known (consider all DAG candidates), or may be incomplete; the method does not account for synergistic or antagonistic actions.

***Causal diagrams for epidemiologic research*
S. Greenland et al. (1999)**

Earlier article on the same topic.

***Using directed acyclic graphs to guide analyses of neighbour health effects: an introduction*
N.L. Fleischer and A.V. Diez Roux (2008)**



Algorithms to decide whether a set of confounders is sufficient, necessary, minimally sufficient.

For implementations, check: DAGitty (browser-based), Tetrad (Java) or dagR (undocumented).

***Adjustment criteria in causal diagrams: an algorithmic perspective*
J. Textor and M. Liškiewicz (2011)**

Efficient algorithm.

***A tool for filtering information in complex systems*
M. Tumminello et al. (2005)**

The minimum spanning tree (take the edges one by one, starting with the most important ones, and keep them if they do not introduce a loop) discards too much information: instead of looking for an acyclic graph, one can look for planar graphs (giving the “planar maximally filtered graph”, PMFG) or, more generally, graphs of genus g . Instead of highlighting edges, this highlights 3- and 4-cliques,  and .

***Spread of risk across financial markets: better to invest in the peripheries*
F. Pozzi et al. (2013)**

Stocks in the periphery of the minimum spanning tree (MST) or the planar maximally filtered graph (PMFG) provide more diversification benefit. Those graphs can be used to represent the contents of a portfolio. Try a long-short strategy, periphery vs center, with minimum variance portfolios on each side, using the following measures of centrality: degree, betweenness, eccentricity, closeness, eigenvector centrality.

`import sage.graphs.genus`

To compute the genus of a graph $G = (V, E)$, consider all possible *combinatorial embeddings*, i.e., all possible cyclic orderings at each vertex. Formally, a combinatorial embedding is an action of the free group $\mathbf{Z} * \mathbf{Z} = \langle v, e \rangle$ on the set of half-edges, with the orbits for $\langle v \rangle$ corresponding to the vertices and those for $\langle e \rangle$, to the edges (e is an involution). The genus of the embedding is given by $2 - 2g = V - E + F$, where V , E and F are the number of orbits for v , e and ve .

***Determining the genus of a graph*
A. Perez (2009)**

A graph is planar iff it does not contain (a subdivision of) $K_5 = \star$ or $K_{3,3} = \bowtie$ (Kuratowski theorem).

The *genus* of a graph is the minimum g such that the graph can be embedded in a surface of genus g .

The *maximum genus* of a graph is the maximum g such that the graph can be embedded in a surface of genus g and, cutting the surface along the graph gives 2-cells, i.e., pieces homeomorphic to the open disk.

***The use of correlation networks in parametric portfolio policies*
H. Lohre et al. (2014)**

Nothing new: the authors suggest to look at the minimum spanning tree (MST) built from the correlation matrix of stock or sector returns, and look at graph metrics such as

- The graph diameter (a small stock network diameter, or a *large* sector network diameter, is a sign of market instability);
- The betweenness centrality, which can be used in a parametric portfolio (a parametric portfolio is a portfolio whose weights are a linear combination of some investment signals: since only the coefficients have to be estimated, the quadratic optimization problem is much simpler).

***Finding community structure in very large networks*
A. Clauset et al.**

To find communities in a graph, one can use a greedy agglomerative algorithm: start with each node in its own cluster; merge the two clusters that produce the largest increase in modularity (the *modularity* is the difference between the fraction of edges inside clusters and the expected fraction); iterate. Instead of keeping track of the adjacency matrix A_{ij} as you merge the clusters (it is sparse, but finding the pair that increases the modularity the most creates dense matrices), keep track of ΔQ_{ij} , the increase in modularity if we merge i and j .

***Quadrature rule-based bounds for functions of adjacent matrices*
M. Benzi and P. Boito (2010)**

Many graph metrics can be computed using matrix functions (often, the exponential) and the adjacency matrix A , e.g.:

- Degree;
- Subgraph centrality of a node i : $[e^A]_{ii}$;
- Estrada index, $EE(G) = \sum e^{\lambda_k} = \text{tr } e^A$;
- Communicability between i and j , $[e^A]_{ij}$; average communicability from i ;
- Communicability betweenness of k :

$$\sum_{ijk \text{ different}} \frac{[e^A]_{ij} - [e^{A-E_k}]_{ij}}{[e^A]_{ij}}$$

where E_k is the adjacency matrix of the graph without the edges linked to node k ;

- Resolvent subgraph centrality, etc.: replace e^x with

$$f(x) = \left(1 - \frac{x}{N-1}\right)^{-1}.$$

They can be estimated by Gaussian quadrature (*i.e.*, $\int_{-1}^1 f(x)dx \approx \sum w_i f(x_i)$; choose n and w_i ; compute the optimal x_i).

Ranking hubs and authorities using matrix functions M. Benzi et al.

Graph metrics defined by matrix functions applied to the adjacency matrix A can be generalized to directed graphs by considering the undirected bipartite graph $\begin{pmatrix} 0 & A \\ A' & 0 \end{pmatrix}$. The node metrics then have two variants: the hub and authority ones.

Functions of matrices N.J. Higham (2005)

A complex function f can be evaluated on a matrix A as follows.

- If $f(z) = \sum a_n z^n$ has convergence radius R , and $\text{Spec } A \subset B(0, R)$, then $f(A) = \sum a_n A^n$.
- If f is \mathcal{C}^∞ on a neighbourhood of $\text{Spec } A$, and $Z^{-1}AZ = \text{diag}(J_1, \dots, J_p)$ is the Jordan decomposition of A , then $f(A) = Z \text{diag}(f(J_1), \dots, f(J_p))Z^{-1}$, where

$$J_k = \begin{pmatrix} \lambda & 1 & & 0 \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ 0 & & & \lambda \end{pmatrix}$$

$$f(J_k) = \begin{pmatrix} f(\lambda) & f'(\lambda) & \dots & \frac{f^{m-1}(\lambda)}{(m-1)!} \\ & f(\lambda) & \ddots & \\ & & \ddots & f'(\lambda) \\ 0 & & & f(\lambda) \end{pmatrix}$$

- If $\lambda_1, \dots, \lambda_s$ are the eigenvalues of A , n_1, \dots, n_s the sizes of the largest Jordan blocks for each of them, r the polynomial of degree at most $\sum n_i$ such that

$$\forall i \in [1, s] \quad \forall j \in [0, n_i - 1] \quad r^{(j)}(\lambda_i) = f^{(j)}(\lambda_i),$$

then $f(A) = r(A)$.

- $f(A) = \frac{1}{2\pi i} \oint_{\Gamma} f(z)(zI - A)^{-1} dz$, where f is analytic inside Γ and $\bar{\Gamma}$ encloses $\text{Spec } A$.

Common examples include exponential, logarithm, sine, cosine, square root (not always defined), sign (only defined if $i\mathbf{R} \cap \text{Spec } A = \emptyset$).

Numeric methods include:

- Polynomial or rational approximations (for exp, log, sin, cos: use Padé's approximation and scaling);
- Factorization: write $A = X^{-1}BX$, where $f(B)$ is easier to compute; for instance, if B is upper triangular, the diagonal of $f(B)$ is $f(b_{ii})$, the coefficients immediately above the diagonal can be computed by solving $f(B)B = Bf(B)$, and one can iterate for the rest (Parlett recurrence – if only works if the eigenvalues are different, but there is a block variant if they are not).
- Iterative methods, $X_{n+1} = g(X_n)$, often coming from Newton's method for some equation satisfied by $f(z)$ (but beware of numeric instability). For instance,

$$X_0 = A$$

$$Y_0 = I$$

$$X_{k+1} = \frac{1}{2}(X_k + Y_k^{-1})$$

$$Y_{k+1} = \frac{1}{2}(Y_k + X_k^{-1})$$

for the square root or

$$X_0 = A$$

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-1})$$

for the sign.

Quantitative easing and volatility spillovers across countries and asset classes Y. Zhou

One can use contemporaneous causal inference to study the relations between various volatility indices.

Dynamical macro-prudential stress testing using network theory S. Levy-Carciente et al. (2014)

There are two main channels of risk contagion:

- Direct interbank liability linkages;
- Changes in bank asset values.

One can account for the latter with a bipartite bank-asset network.

**Global portfolio investment network
and stock market co-movement**
T. Chuluun (2014)

Cross-border capital flows (data from the IMF CPIS (coordinated portfolio investment survey), 2001–present) define a network of countries. Central countries (high degree, indegree (better), outdegree (worse) or eigenvector centrality) are more correlated with global indices. Other flows define other networks: trade flows, foreign direct investment, bank loans, etc.

Stochastic flow diagrams
N.J. Calkin and M. López de Prado (2014)

The authors have re-invented graphical models and/or recurrent neural networks, with a slightly different graphical notation.

The topology of macro financial flows
N.J. Calkin and M. López de Prado (2014)

Instead of building a minimum spanning tree (or more general graph) from a correlation matrix, one can build a graphical model from an SVAR(p) model.

**Structural analysis
with independent innovations**
H. Herwartz (2014)

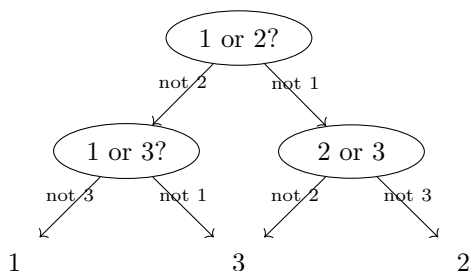
Often, multi-variate time series models (e.g., SVAR, structural vector autoregressive model) are not identified: independent component analysis (ICA) can help.

**Emergence of statistically validated financial
intraday lead-lag relationships**
C. Curme et al. (2014)

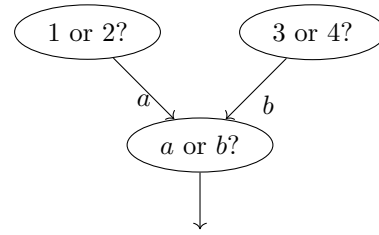
Compute the cross-correlation matrix of stock returns, convert the correlations to p -values, adjust them for multiple testing, truncate them, and build the corresponding graph. As the lag increases (from 5 minutes to 1 hour), the graph becomes sparser.

**Optimizing directed acyclic graph
support vector machines**
F. Takahashi and S. Abe

When using support vector machines (SVM) on multiclass problems with the one-against-all or the pairwise approach, there are unclassified regions. To avoid them, one can use a smaller number of pairwise models, arranged in a tree, either by eliminating one element at each level of the tree



or using a tournament,



But the performance depends on the directed acyclic graph (DAG) chosen. It is better if easy-to-classify class pairs (low support vectors/observations ratio) appear in the top layers of the graph, and the hard-to-classify ones in the leaves.

**A comparison of methods
for multiclass support vector machines**
C.W. Hsu and C.J. Lin (2002)

Among the multiclass SVM algorithms, prefer the pairwise or DAG ones to the (more complicated) “all together” methods.

Algorithms for the multi-armed bandit problem
V. Kuleshov and D. Precup (2000)

Simple multi-arm bandit algorithms (to manage the exploration-exploitation trade-off in *reinforcement learning*) work well:

- ε -greedy: select the arm with the highest empirical mean reward $\hat{\mu}_i$ with probability $1 - \varepsilon$, and a random arm with probability ε ;
- Boltzmann: $p_i \propto \exp(\hat{\mu}_i/T)$, where the temperature T slowly decreases;
- Pursuit:

$$p_i \leftarrow \begin{cases} (1 - \beta)p_i + \beta \cdot 1 & \text{if } i = \text{Argmax}_j \hat{\mu}_j \\ (1 - \beta)p_i + \beta \cdot 0 & \text{otherwise;} \end{cases}$$

- Reinforcement comparison:

$$\begin{aligned} p_i &\propto \exp \pi_i \\ \pi_j &\leftarrow \pi_j + \beta(r - \bar{r}) \\ \bar{r} &\leftarrow (1 - \alpha)\bar{r} + \alpha r \end{aligned}$$

where

$$\begin{aligned} \pi_i &= \text{preference} \\ j &= \text{arm selected} \\ r &= \text{reward} \\ \bar{r} &= \text{average reward;} \end{aligned}$$

- Upper confidence bound

$$j = \text{Argmax}_i \left(\hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}} \right)$$

where

$$\begin{aligned} \hat{\mu}_i &= \text{average reward for arm } i \\ n_i &= \text{number of times } i \text{ was chosen} \\ t &= \sum n_i = \text{number of steps.} \end{aligned}$$

Applications include clinical trials and online advertising.

***Hierarchical temporal memory
including HTM cortical learning algorithms***
Numenta (2011)

Recurrent (convolutional?) deep neural network, with connections *within* layers, binary weights, and sparsity constraints.

A survey on transfer learning
S.J. Pan and Q. Yang (2010)

Transfer learning studies what to do when the training and test sets are significantly different. For instance: how to extend a document classification model to a new domain (say, a sentiment model, from camera reviews to hotel reviews)? How to deal with a time-changing domain? Some new learning will be needed, but we want to reuse as much information as possible – we want to automatically identify the bits of the model that can be kept and those that should be discarded. Approaches include:

- Instance transfer: reweight the source data (Ada-boost, importance sampling);
- Feature transfer: reweight/select the features to reduce the distance between the source and the target domains;
- Semisupervised learning;
- Use the source model as a prior for the target one.

Markov logic networks (MLN) and inductive logic programming (ILP) were also mentioned, with no details.

MEG decoding across subjects
E. Olivetti et al. (2014)

Application of transfer learning (instance transfer, *i.e.*, importance sampling): decoding an MEG when training and test data are from different subjects.

***Correlation and large-scale simultaneous
significance testing***
B. Efron

In presence of high correlation, the p -values are still $U(0, 1)$ under H_0 , but not iid: their sample distribution can be: wider (if many correlations are large and negative); narrower and skewed (if many correlations are large and positive).

***High-dimensional variable selection
for survival data***
H. Ishwaran et al. (2010)

To gauge the importance of a variable in a random forest model, look at (the distribution of) its minimum depth.

***A factor model to analyze heterogeneity
in gene expression***
Y. Blum et al. (2010)

Approaches to multiple testing often assume independence – FAMT (factor analysis for multiple testing) relaxes that assumption [no details]. In R, check the FAMT package.

***Measuring and testing dependence
by correlation of distances***
G.J. Székely et al. (2008)

Original article on distance covariance

$$V^2(X, Y) = \int_{\mathbf{R}^{p+q}} \frac{|f_{XY}(t, s) - f_X(t)f_Y(s)|^2}{\|t\|_p^{1+p} \|s\|_q^{1+q}} dt ds.$$

***A non-iterative bayesian approach
to statistical matching***
S. Rässler (2003)

Statistical matching (aka “data fusion”) can be seen as a missing data problem, and tackled by *stochastic regression imputation* (NIBAS): compute the regressions $X \sim Z$, $Y \sim Z$ (where Z are the common variables), Σ_X , Σ_Y , $\text{Cor}(X, Y)$ (use the identity if you have no information), and sample Σ , β_X , β_Y , $X|y, \beta, \Sigma$, $Y|x, \beta, \Sigma$.

***Coherent mortality forecasting:
the product-ratio method
with functional time series***
R.J. Hyndman et al. (2012)

Forecasts (using functional PCA, and time series models for the coefficients of the principal components) of the mortality rates of two subpopulations (say, males and females) often diverge. Instead, forecast the sum and the difference of the (log) rates (if the variances are equal, they are uncorrelated: $\text{Cov}(X + Y, X - Y) = \text{Var } X - \text{Var } Y$). For more than two populations, consider the average (of the log-rates) and the difference with the average.

***Forecasting elections
with non-representative polls***
W. Wand et al. (2014)

Non-representative polls are just stratified surveys: after adjustment, they are perfectly usable.

Super Learner
M.J. van der Laan et al. (2007)

Use *cross-validation* to choose among various models. Besides the usual models (GLM, GLMnet, GAM, etc.), the **SuperLearner** R package also wraps DSA (deletion, substitution, addition), **LogicReg** (logic regression – “logic”, not “logistic”), **polsspline** (adaptive regression splines, aka MARS).

***Subsemble: an ensemble method
for combining subset-specific algorithms fits***
S. Sapp et al. (2013)

Bagging fits models on samples of the data, and averages them. Instead of an unweighted average, one can use cross-validation to find the best weights.

Bayesian estimation supercedes the t-test
M. Meredith and S. Kruschke (2014)

The BEST package replaces the T test with Bayesian statistics (with a Student T distribution to allow for outliers).

```
library(BEST)
r <- BESTmcmc(y1,y2)
r
summary(r)
plot(r)
plotPostPred(r)
s <- r$sigma1^2 / r$sigma2^2
hdi(s)
plotPost(s)
```

***The OpenCPU system: towards a universal
interface for scientific computing through
separation of concerns***
J. Ooms (2014)

OpenCPU is a REST interface (with a well-defined API, inviting other implementations) to R; it is stateless (function results are stored on the server, and a key to access them is returned to the client).

***The stringdist package
for approximate string matching***
M.P.J. van der Loo

The `stringdist` package implements many string distances: Hamming (limited to same-length strings), longest common substring, Levenstein (number of insertions, deletions, substitutions), optimal string alignment (idem, with transpositions of adjacent characters – not a distance), Damerau-Levenstein (idem, but corrected to make it a distance), n -gram distance (L^1 distance between the n -gram count vectors), cosine distance ($1 - \langle u, v \rangle$ is not a distance on the sphere – but $\arccos \langle u, v \rangle$ is), Jaccard distance (number of common n -grams divided by the number of n grams), heuristics (Jaro and Jaro-Winkler distances count “matches” and penalize “non-matches”).

***ScagExplorer: exploring scatterplots
by their scagnostics***
T.N. Dang and L. Wilkinson

To examine a large number of scatterplots, one can summarize them with a small number of metrics (outlying, skewed, clumpy, sparse, striated, convex, skinny, stringy, monotonic), look at the resulting parallel-plot-with-violins and/or cluster them. The authors suggest

the *leader algorithm* (start with an empty leader list and add a new scatter plot if the nearest leader is beyond some Euclidian distance threshold; only display the leaders, or only display the scatterplots attached to a given leader).

***New approaches in visualization
of categorical data: R package extracat***
A. Pilhöfer and A. Unwin (2013)

If you have more than two qualitative variables, check:

```
# Relative multiple barchart
extracat::rmb( ~ V1 + V2 + V2 )
# Categorical parallel coordinate plot
extracat::cpcp(d)
vcd::mosaic(xtabs( freq ~ V1 + V2 + V2 ))
vcd::doubledecker( xtabs(...) )
```

The `extracat::optile` function, used by `cpcp` to re-order the levels of the qualitative variables to reduce overplotting, may also be of interest.

Rainbow color map (still) considered harmful
D. Borland and R.M. Taylor (2007)

It is not perceptually uniform, it is not naturally ordered, its luminance does not vary monotonically.

***RnavGraph: an R package to visualize
high-dimensional data using graphs as
navigational infrastructure***
A. Waddell and W. Oldford (2013)

Not unlike `ggobi`: move from one (2D) scatterplot to another by a rotation (in 3D if they share a variable, in 4D otherwise).

***Dealing with stochastic volatility in time series
using the R package stochvol***
G. Kastner

Bayesian estimation of stochastic volatility models.

***Parallel computing for data science,
with examples in R and beyond***
N. Matloff (2013)

This book reviews, with detailed examples, the main types of parallel systems, mostly in R (clusters, multi-core systems, GPUs):

- `parallel::clusterApply`, `parallel::clusterApplyLB` (with load balancing);
- `parallel::mclapply`;
- `foreach`, `%dopar%`;
- `Rmpi` (low-level programming on clusters, via message passing);
- `Rdsm` (based on `bigmemory`): shared memory (you have to manage the locks yourself);
- `OpenMP`, to automatically parallelize loops in C (yes, it is that easy);
- `TBB` (Threads building blocks): thread-safe C++ containers, with a few parallel algorithms;

- Cilk++: adds `spawn` and `sync` keywords to C (I am not sure how relevant this is, given that C++ now has futures);
- `gputools`
- Thrust: similar to TBB, for the GPU (it can also use OpenMP or TBB if you do not have a GPU);
- Rth: to use Thrust from R;
- OpenBLAS.

(The author wrote Rdsml and Rth.)

A comparison of programming languages in economics
S.B. Aruoba (2014)

Julia, Python+Numba, Matlab+MEX, Mathematica have speeds comparable to C++/Fortran/Java, but functional languages (OCaml, Haskell, Scala) are worth a look.

A reliable effective terascale linear learning system
A. Agarwal et al. (2013)

Most machine learning algorithms can be implemented on a *data-centric computing platform* such as Hadoop using the *AllReduce* operation (arrange the nodes in a tree, use it to compute sums or weighted averages, *i.e.*, to aggregate the partial results), which has much less overhead than MapReduce. However, their implementation bypasses Hadoop's MapReduce (the nodes communicate between themselves via TCP), which may lead to reliability issues (do not expect the system to run for more than a few hours without failures).

Asynchronous functional reactive programming for GUIs
E. Czaplicki and S. Chong (2013)

FRP is programming with signals, *i.e.*, time-changing values (in Haskell, that would be a functor, `fmap: (a → b) → Signal a → Signal b`). Most FRP languages assume the signal changes continuously, and sample from it. Elm (a Haskell DSL) works with discrete signals (asynchronous FRP) and therefore avoids needless recomputations.

MDF

<https://github.com/ahlmss/mdf> (2013)

Data-flow programming in Python.

Clash of the lambdas
A. Biboudis et al. (2014)

Lambdas in Java 8 are mature and efficient. Scala is inefficient, *e.g.*, because of garbage collection or boxing/unboxing (collections are not specialized for primitive types). Optimizing frameworks (ScalaBlitz) help, but not enough: the gap is reduced from 2 orders of magnitude to 1.

Graph drawing in Tikz
T. Tantau (2013)

There are now graph layout algorithms in Tikz, implemented in Lua.

Managing schema evolution in NoSQL data stores
S. Scherzinger et al. (2013)

Schema migration in schema-less databases (NoSQL) are often ad hoc (dangerous, untested) data transformations, either eager (convert everything in one go) or lazy (old and new entries cohabit, they are only converted when accessed). A system-agnostic data migration language would be useful.

Locality-sensitive hashing scheme based on p-stable distributions
M. Datar et al. (2004)

To solve the *k*-nearest neighbour problem probabilistically, in high dimension:

- Find hash functions with a higher probability of collision when the points are close:

$$\text{distance} \leq r_1 \implies P[\text{collision}] \geq p_1$$

$$\text{distance} \geq r_2 \implies P[\text{collision}] \leq p_2;$$

- Apply many such hash functions to the data and examine the points with the most collisions.

Such hash functions can be obtained by projecting the points in a random direction and binning them:

- Choose $r > 0$;
- Pick $b \sim U(0, 1)$ and $a \sim F$, where F is an n -dimensional p -stable distribution;
- Let $h_{a,b}(v) = \left\lfloor \frac{a \cdot v + b}{r} \right\rfloor$.

A distribution F is p -stable if, for all n , for all $v_1, \dots, v_n \in \mathbf{R}$, if $X_1, \dots, X_n, X \sim F$ are iid, then $\sum_i v_i X_n$ and $(\sum |v_i|^p)^{1/p} X$ have the same distribution. The Cauchy distribution is 1-stable; the Gaussian distribution is 2-stable.

The development of hyperdual numbers for exact second-derivative calculations
J.A. Fike and J.J. Alonso (2011)

The first derivative of a function f is often approximated using

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h)$$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2),$$

but this is not numerically stable: there is a *subtractive cancellation error* problem, which prevents h from being too small (often, 10^{-4} to 10^{-6} is the best we can do). The complex step approximation

$$f'(x) = \frac{\text{Im } f(x+ih)}{h} + O(h^2)$$

addresses this problem: h can be very small (10^{-14}). But the problem reappears if you want the second derivative.

Instead of complex numbers, one can use other extensions of \mathbf{R} , such as the dual numbers $\mathbf{R}[\varepsilon]/(\varepsilon^2)$,

$$f(x + \varepsilon) = f(x) + f'(x)\varepsilon$$

or the *hyperdual numbers*, $\mathbf{R}[\varepsilon, \eta]/(\varepsilon^2, \eta^2, \varepsilon^2\eta^2)$,

$$f(x + \varepsilon + \eta) = f(x) + f'(x)\varepsilon + f'(x)\eta + f''(x)\varepsilon\eta.$$

The derivatives are exact, and there is no step size to choose.

That is easy to implement in languages with operator overloading (C++, Julia), but 10 times slower than the difference method.

***Unbounded spigot algorithms
for the digits of pi***
J. Gibbons (2005)

Good job interview question: implement a streaming algorithm to compute the digits of π , ad infinitum, using

$$\pi = \sum_{n \geq 0} \frac{(n!)^2 2^{n+1}}{(2n+1)!}.$$

The formula comes from

$$\frac{\pi}{4} = \arctan 1 = \int_0^1 \frac{dx}{1+x^2} = \sum \frac{(-1)^n}{2n+1}.$$

The series converges very slowly, but it is alternating, and taking the average of two consecutive partial sums gives a faster converging series – but it is still alternating, so we can use the same trick, again and again. This is the *Euler convergence acceleration transform*.

This transformation can also be motivated as follows: given

$$f(x) = \sum_{n \geq 0} (-1)^n a_n x^n,$$

show that

$$g(y) = f\left(\frac{y}{1-y}\right) = \sum_{n \geq 0} (-1)^n \Delta^n a_0 y^n,$$

where Δa is the forward difference of a , and compute $g(\frac{1}{2}) = f(1)$.

Here is a straightforward algorithm: find an upper and lower bound on the error,

$$s_n - \ell_n \leq \pi \leq s_n + u_n,$$

and use it to infer if the k th decimal of s_n is that of π – for this, you need arbitrary precision rationals.

***Computing extremely accurate quantiles
using t-digests***
T. Dunning

To compute a quantile of a datastream, approximately, one can:

- Sample from it, online, and take the quantile of the sample – for extreme quantiles, you only need to keep a sample of the tails;
- Bin the data into a set of centroids and counts, and collapse the centroids when there are too many of them (*t-digest*).

***The PH-tree – a space-efficient storage
structure and multi-dimensional index***
T. Zäschke et al. (2014)

Indexes for spacial data are often KD-trees (split the space in half, along an axis-aligned hyperplane, at each node), quad-trees (split the space in 2^n at each node) or R-trees (trees of bounding boxes). One can also use *prefix trees* (tries), by interleaving the different dimensions into a single bitstring. There is no need for rebalancing.

Nepotism and equity prices
G. Zhu (2014)

Nepotism (several family members being directors in the same company – in the US, this has to be reported in text form; the information can be extracted by looking for keywords such as “brother”, “father”, etc., used to build a social network and compute the usual network metrics) is detrimental to firm value, except when there is a large outside block holder.

Textual classification of SEC comment letters
J.P. Ryans (2014)

SEC comment letters are a good predictor of future negative abnormal returns (use a naive Bayes classifier), especially if they have been accessed by many investors (number of downloads).

***Efficient online summarization
of microblogging streams***
A. Alariu (2014)

A *word graph* (the nodes are words, the edges are bigrams, with weights) can be used to summarize a corpus (look for the highest-weight path). One can also use trigrams and let the weights decay (multiply by $1 - \varepsilon$ at each step).

***Looking for risk in words:
a narrative approach to measuring
the pricing implications of finance constraints***
M.M.M. Buehlmaier and T.M. Whited (2014)

Financially constrained firms (those having difficulty raising funds) can be identified from the text in the MD&A section of their 10-K filing, using the naive

Bayes algorithm (building the training set is problematic, though).

Finance constraints
M.M.M. Buehlmaier (2014)

One can estimate how financially constrained a company is by the textual analysis of its regulatory filings, using a Naive Bayes classifier. The training sets are questionable:

- A small, subjective manually-generated training set;
- Companies identified by a set of words known to be associated with delays resulting from the difficulty of issuing new equity;
- Idem for debt.

One could also check for the effect of pairs, triplets, quadruplets of words.

The price of silence: when no one asks questions during conference calls
S. Chen et al. (2014)

The number of questions asked by investors in earnings-related conference calls measures the quality of a firm's communication.

News-implied volatility and disaster concerns
A. Manela and A. Moreira (2013)

Use SVM regression to predict the VIX from front-page articles of the Wall Street Journal.

Portfolio selection with options and transaction
S. Malamud (2014)

Introducing options in your investment universe dramatically increases the number of assets, and the variance matrix becomes untractable – and, option traders would argue, largely irrelevant. Instead of a mean-variance portfolio, one can look for greek-efficient portfolios – with a high value for the greeks you want, and a small value for those you do not. [I would still use the mean-variance framework, and include the greeks as constraints, setting those we want to ± 1 , and those we do not to 0.]

Volatility and directional information-based trading in options

Generalization of the PIN model to option markets, accounting for two types of information, direction and volatility.

Why does the option to stock volume ratio predict stock returns?
L. Ge

Most option-implied measures only use the price: the volume, normalized by the stock volume, and decomposed into call/put, buy/sell, open/close, is also a

predictor of future returns. Opening and closing of long call positions are more informative than synthetic shorts.

CDS inferred stock volatility
B. Guo (2014)

Since both CDSs and OTM puts can be used to protect against downside risk, one can define a CDS-implied volatility. It is different from the option-implied one, and is a better predictor of future stock volatility.

Option listing and information asymmetry
J. Hu (2014)

Option listing improves market efficiency and reduces information asymmetry. (The study uses propensity score matching.)

Stock market ambiguity and the equity premium
A. Kagkadis (2014)

The dispersion of volume-weighted strike prices is a measure of market ambiguity (high ambiguity leads to high hedging demand with different beliefs, which leads to a higher dispersion of strikes). It is related to, but different from, other option-implied quantities (variance risk premium (VRP), smirk slope, risk-neutral variance, skewness, kurtosis, open-interest ratio (OTM puts/ATM calls, a measure of hedging pressure)). It complements the VRP. (Analyst forecast dispersion is another measure of ambiguity.)

Wealth transfers via equity transactions
R.G. Sloan and H. You (2013)

The option of a company to issue new shares or buy back shares when its stock is over- or under-priced should be included in its valuation. It corresponds to a wealth transfer from new to old investors.

Demand for crash insurance, intermediary constraints and stock return predictability
H. Chen et al. (2013)

New deep out-of-the-money S&P 500 put options are a good (negative) predictor of future (monthly) returns.

An option to cheat: an application of option theory to realize flipping in underpricing
J. Stojkovic (2013)

IPO flipping is the sale of an (undervalued) IPO stock on day one if/when the price jumps. It can be modeled as a perpetual American call option, with the IPO price as strike. The option price can be decomposed into the potential immediate profit, and the time value of waiting.

Does VIX truly measure return volatility?

K.V. Chow et al. (2014)

The current VIX computations (price of a portfolio of OTM put or call options, with weights inversely proportional to the square of the strike) are not really model-free (they assume a diffusion). It is a biased estimator of expected volatility, but it can be corrected.

Merton's model, credit risk and volatility skews

J. Hull et al. (2004)

Merton's model assess the credit risk of a company by considering its equity as an option on the assets – the spot is the value of the assets, the strike is the value of the debt, the expiry its maturity. Options on equity are therefore *compound options* on the assets – this explains the skew. One can use the option market to estimate the model parameters and compare the results with CDS spreads.

Where would the EUR/CHF exchange rate be without the SNB's minimum exchange rate policy?

M. Hanke et al. (2013)

Capped exchange rates (e.g., the floor on EUR/CHF) can be modeled as American options on the latent (uncapped) FX.

Option-implied volatility measures and stock return predictability

X. Fu et al.

The following option-derived quantities have some predictive power on future stock returns:

$$\begin{aligned} \text{IVSpread} &= \text{IV}_{\text{ATM, Call}} - \text{IV}_{\text{ATM, Put}} \\ \text{IVSkew} &= \text{IV}_{\text{OTM, Put}} - \text{IV}_{\text{ATM, Call}} \\ \text{AMB} &= \frac{1}{2}(\text{IV}_{\text{ITM, Put}} + \text{IV}_{\text{ATM, Call}}) - \frac{1}{2}(\text{IV}_{\text{ITM, Call}} + \text{IV}_{\text{ATM, Put}}) \\ \text{COMA} &= \text{IV}_{\text{OTM, Call}} - \text{IV}_{\text{ATM, Call}} \\ \text{POMA} &= \text{IV}_{\text{OTM, Put}} - \text{IV}_{\text{ATM, Put}} \\ \text{RVIV} &= \text{RV} - \frac{1}{2}(\text{IV}_{\text{ATM, Put}} + \text{IV}_{\text{ATM, Call}}). \end{aligned}$$

Jump and variance risk premia in the S&P 500

M. Neumann et al. (2014)

By estimating the physical and risk-neutral (option prices) SVCJ (stochastic volatility with contemporaneous jumps in returns and variance) model,

$$\begin{aligned} ds &= \mu dSdt + S\sqrt{V}dW_1 + Z_1 dN \\ dV &= \kappa(\theta - V)dt + \sigma\sqrt{V}dW_2 + Z_2 dN, \end{aligned}$$

one can define the following risk premia:

- Diffusive variance risk premium: $\kappa^{\mathbf{Q}} - \kappa^{\mathbf{P}}$;
- Price jump risk premium: $\mu^{\mathbf{Q}} - \mu^{\mathbf{P}}$;
- Variance jump risk premium $\theta^{\mathbf{Q}} - \theta^{\mathbf{P}}$.

They are related to macroeconomic uncertainty.

Risk-adjusted option-implied moments

F. Brinkmann and O. Korn (2014)

The (model-free) risk-neutral density can be computed from option prices; the physical density (and the physical moments) can be computed in the same way, using physical expected discounted payoffs instead of prices. Those physical prices can be computed from market prices and the utility function of a representative investor. A CRRA utility function can explain the variance risk premium (VRP),

$$\text{VRP} \stackrel{\text{def}}{=} E^{\mathbf{P}}[\text{variance}] - E^{\mathbf{Q}}[\text{variance}],$$

but not both the VRP and the skewness risk premium.

Pricing interest rate derivatives and computing pathwise Greeks in the extended Libor market model

T. Lee (2013)

There is no longer a single yield curve; the Libor market model (LMM) can be extended to this new multi-curve environment.

Expected bond returns and the credit risk premium

Z. Afik and S. Benninga (2014)

Do not confuse yield to maturity (computed from the promised cash flows) and expected bond returns (computed from the expected cash flows, *i.e.*, accounting for rating, rating transition matrix and industry recovery rate). The difference between the two, the credit risk premium, is (empirically) a simple function of rating and duration.

Asymmetry in stock returns: an entropy measure

L. Jiang et al. (2014)

Asymmetry tests usually rely on exceedance correlations

$$\begin{aligned} \rho_c^+ &= \text{Cor}(X, Y \mid X, Y \geq c) \\ \rho_c^- &= \text{Cor}(X, Y \mid X, Y \leq -c). \end{aligned}$$

Tests based on exceedance densities

$$\begin{aligned} f_c^+(x, y) &= f(x, y, \mid x, y \geq c) \\ f_c^-(x, y) &= f(x, y, \mid x, y \leq -c) \end{aligned}$$

and the “entropy”

$$S_c = \frac{1}{2} \iint_{RR^2} \left(\sqrt{f_c^+} - \sqrt{f_c^-} \right)^2$$

are more powerful.

Introduction to nonextensive statistical mechanics and thermodynamics
C. Tsallis et al. (2003)

Tsallis entropy (or q -entropy, or non-extensive entropy) generalizes entropy.

The exponential function e^x is a solution of $y' = y$; its inverse, $\ln x$, is additive: $\ln(xy) = \ln x + \ln y$; the entropy is

$$S = \sum_i p_i \ln \frac{1}{p_i} = \left\langle \ln \frac{1}{p_i} \right\rangle.$$

The q -exponential e_q^x is a solution of $y' = y^q$; its inverse,

$$\ln_q x = \frac{x^{1-q} - 1}{1 - q},$$

is pseudo-additive:

$$\ln_q(xy) = \ln_q x + \ln_q y + (1 - q) \ln_q x \ln_q y;$$

the q -entropy is

$$S_q = \sum_i p_i \ln_q \frac{1}{p_i} = \left\langle \ln_q \frac{1}{p_i} \right\rangle.$$

The characterizations of entropy (Shannon or Khinchin theorem) generalize to q -entropy, with one more term or exponent, e.g.,

$$S(A+B) = S(A) + S(B) + (1-q)S(A)S(B) \text{ if } A \perp\!\!\!\perp B$$

$$S(A+B) = S(A) + S(B|A) + (1-q)S(A)S(B|A)$$

Tsallis entropy is related to other generalized entropies, e.g.,

$$S_q^{\text{Renyi}} = \frac{1}{1-q} \ln \sum_i p_i^q = \frac{\ln[1 + (1-q)S_q]}{1-q}.$$

Realized coskewness of the VIX and S&P 500 and the equity premium
Z. Liu et al. (2014)

In factor models, besides volatility, idiosyncratic volatility, $\beta(\text{stock}, \text{market})$, one can also look at coskew(stock, market) (but it can be tricky to estimate – if you have high-frequency data, you can try the realized skewness). Coskew(VIX, S&P 500) is related to, but different from, the variance risk premium (VRP).

Which beta is best? On the information content of option-implied beta
R. Baule et al. (2014)

One can define many option-implied betas:

- Using implied volatility (IV) and historical correlation;
- Using IV and assuming constant correlation;
- Using IV and assuming some specific option-pricing model;

- Using IV and cross-correlation derivatives (if available);
- Using the implied moments: if $y = \alpha + \beta x + \varepsilon$, then

$$\beta = \sqrt{1 - \frac{\text{Var } \varepsilon}{\text{Var } y}} \left(\frac{\text{Var } y}{\text{Var } \varepsilon} \right)^{1/2} = \left(\frac{\text{Skew } y}{\text{Skew } \varepsilon} \right)^{1/3}$$

Option-implied betas based on the variance, and those that ensure that the market beta is one (for this, jointly estimate the betas for all the stocks) are more accurate.

Extremal dependence and contagion
R. Fry-McKibbin and C.Y.L. Hsiao (2014)

During crises, not only do the moments change (returns drop, volatility jumps), so do the *comoments*: for instance, $\text{Cor}(\text{ret}, \text{vol})$ changes sign, from negative (volatility skew) in normal times, to positive (volatility smile) in crises. One can devise *contagion tests* by looking at $\text{Cor}(\text{ret}_i, \text{vol}_j)$, $\text{Cor}(\text{ret}_i, \text{skew}_j)$, $\text{Cor}(\text{vol}_i, \text{vol}_j)$.

Modelling dependence structure and forecasting portfolio value-at-risk with dynamic copulas
M. Cerrato et al. (2014)

Look at the asymmetric dependence, e.g.,

- Threshold correlation,

$$\rho^- = \text{Cor}(X_1, X_2 \mid X_1 \leq x_1, X_2 \leq x_2),$$

where x_1, x_2 are the p -quantiles of X_1 and X_2 , and

$$\rho^+ = \text{Cor}(X_1, X_2 \mid X_1 \geq x_1, X_2 \geq x_2),$$

where x_1, x_2 are the $(1-p)$ -quantiles;

- Quantile dependence

$$\lambda^p = P[X_1 \leq x_1 \mid X_2 \leq x_2]$$

or

$$\lambda^p = P[X_1 \geq x_1 \mid X_2 \geq x_2];$$

- Tail dependence

$$\lambda^U = \lim_{p \rightarrow 1} \lambda^p$$

$$\lambda^L = \lim_{p \rightarrow 0} \lambda^p$$

between the long and short portfolios built from the following signals

- $\beta(\text{stock}, \text{market})$;
- $\text{coskew}(\text{stock}, \text{market}, \text{market})$;
- $\text{cokurt}(\text{stock}, \text{market}, \text{market}, \text{market})$.

To estimate the VaR, use the GAS (generalized autoregressive score) model.

Asymmetry in tail dependence of equity portfolios
E. Jondeau

To estimate tail dependence, prefer parametric methods (unless you really have a lot of data), e.g., using the multivariate non-central t distribution.

***Cornish-Fisher versus Gramm-Charlier:
an appraisal***
D. Maillard (2014)

Cornish-Fisher is a transformation of a Gaussian random variable

$$Y = X + (X^2 - 1)\frac{S}{6} + (X^3 - 3X)\frac{K}{24} - (2X^3 - 5X)\frac{S^2}{36};$$

if S and K are small, they are close to the skewness and excess kurtosis of Y .

Gram-Charlier is a transformation of the Gaussian density ϕ :

$$\psi(x) = \phi(x) \left(1 + \frac{S}{6} H_3(x) + \frac{K}{24} H_4(x) \right)$$

$$H_n(x) = (-1)^n e^{-x^2/2} \frac{d^n}{dx^n} e^{-x^2/2}$$

(this time, S and K are really the skewness and excess kurtosis). The domain of validity of the Cornish-Fisher transformation (the set of skewness-kurtosis pairs attainable) is much larger.

***Expansion of probability density functions
as a sum of gamma densities
with applications in risk theory***
N.L. Bowers (1966)

The idea behind the Gram-Charlier expansion of a pdf,

$$p(x) = \phi(x) + \sum_{n \geq 1} \dots = \sum_{n \geq 0} a_n H_n(x) \phi(x),$$

where the H_n are orthogonal polynomials wrt the Gaussian distribution ϕ , can be generalized to other reference distributions, e.g., for positive random variables, the Gamma distribution (and Laguerre polynomials).

***Robust series expansions
for probability density estimation***
M. Welling

Use robust moments

$$\left\langle (\alpha x)^n e^{\frac{1}{2}(1-\alpha^2)x^2} \right\rangle, \quad 1 \leq \alpha^2 \leq 2,$$

to define robust Gram-Charlier or Hedgeworth expansions.

Joining risks and rewards
H.J. Stein (2014)

When you need both the **P** and **Q** measures (e.g., for risk and pricing), use the same samples, but with different weights (*i.e.*, use importance sampling, as when pricing deep OTM options).

***Identifying jumps in financial assets:
comparison between nonparametric jump tests***
A.M. Dumitru and G. Urga (2011)

Review of many jump estimators or tests, e.g., those based on

$$\text{BV} = \sum |r_t r_{t-1}|$$

$$\sum \text{Min}(r_t, r_{t-1})^2$$

$$\sum \text{median}(r_t, r_{t-1}, r_{t-2})^2$$

$$\text{Max} \frac{|r_t|}{\text{BV}}$$

$$\frac{\sum_{t \equiv 0 \pmod{\delta}} |r_t + \dots + r_{t+\delta-1}|^m}{\sum |r_t|^m}$$

$$\sum R_t - r_t \text{ (ratio vs log-returns), etc.}$$

***Monitoring housing markets
for episodes of exuberance***
E. Pavlidis et al. (2013)

The ADF test can be generalized to distinguish between integrated $I(1)$ and explosive behaviours:

$$\Delta y_t = \alpha + \beta y_{t-1} + \sum_{i=1}^k \gamma_i \Delta y_{t-i} + \varepsilon, \quad t \in [T_1, T_2]$$

$$\text{ADF}_{T_1 T_2} = \frac{\beta_{T_1 T_2}}{\sigma(\beta_{T_1 T_2})}$$

$$\text{SADF}_{T_1} = \sup_{T_2 \geq T_1} \text{ADF}_{T_1 T_2} \quad \text{expanding window}$$

$$\text{GSADF}_w = \sup_{T_2 - T_1 \geq w} \text{ADF}_{T_1 T_2} \quad \text{window size at least } w.$$

***A compound multifractal model for
high-frequency asset returns***
E.M. Aldrich et al. (2014)

High-frequency returns (E mini S&P 500 futures contract), in trade space, away from prescheduled news announcements, are Gaussian. Intertrade duration can be modeled with an exponential distribution, conditional on a hidden state (k binary state variables, with varying degrees of persistence).

$$d_t \sim \text{Exp} \left(\lambda \prod_i M_{it} \right)$$

$$M_{i,t+1} = \begin{cases} M_{it} & \text{with probability } 1 - p_i \\ 2 - M_{it} & \text{with probability } p_i \end{cases}$$

$$M_{i0} = m_0 \in (0, 2)$$

A truncated version of this Markov switching multifractal duration (MSMF) model, $\text{Min}(d_t, d_t^0)$, where $d_t^0 \sim \text{Exp}(\lambda^0)$, gives a better fit.

When to sell Apple and the Nasdaq? Trading bubbles with a stochastic disorder model
A. Shiryaev et al. (2014)

Consider a geometric brownian motion with a change point in the trend,

$$dS_t = (\mu_1 \mathbf{1}_{t < \theta} + \mu_2 \mathbf{1}_{t \geq \theta}) S_t dt + \sigma dB_t$$

(the breakpoint, θ , is called a “moment of disorder” in quality control). Close a long (resp. short) position at

$$\tau_{\text{long}} = \underset{\tau \leq T}{\underset{\tau \text{ stopping time}}{\operatorname{Argmax}}} E[S_\tau]$$

$$\tau_{\text{long}} = \underset{\tau \leq T}{\underset{\tau \text{ stopping time}}{\operatorname{Argmin}}} E[S_\tau].$$

It can be estimated explicitly.

Empirical asset pricing: Eugene Fama, Lars Peter Hansen and Robert Shiller
J.Y. Campbell (2014)

Around the notion of stochastic discount factor.

The systematic risk of private equity
A. Buchnet and R. Stucke (2014)

To estimate the risk of a private equity fund, model its dividend payments as a stochastic process.

$$\text{Dividend yield} = 1 - e^{-\delta_t}$$

$$\delta_t = \delta_0 + z_t t$$

$$z_t \sim N(0, \sigma)$$

The valuation of M&A targets by relative indifference pricing
C. Mauch and S. Rostek (2013)

The *indifference price* is the price at which a given investor (initial wealth and utility function) is indifferent between holding and not holding the asset. The buying and selling (*i.e.*, long and short) indifference prices are different. Indifference pricing is relevant for incomplete markets, *e.g.*, for real options.

Return and risk of pairs trading using a simulation-based bayesian procedure for predicting stable ratios of stock prices
L.T. Gatarek et al. (2014)

When testing for cointegration, only the subspace of cointegration relations is well-defined. To single out a cointegration relation, some “normalization” is needed, *e.g.*, $\beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \beta_2 \end{pmatrix}$, or $\beta' \beta = I$, *i.e.*, $\beta = \begin{pmatrix} \beta_1 \\ \sqrt{1 - \beta_1^2} \end{pmatrix}$ (only the leverage changes). There are several rules to choose the size of the position on the spread $s_t = \beta_1 y_{1t} + \beta_2 y_{2t}$, *e.g.*, $-s_t$, $\text{sign } E_t[\Delta s_{t+1}]$ or $E_t[\Delta s_{t+1}] \times \text{sign } E_t[\Delta s_{t+1}]$ or a buy-and-hold strategy.

Financial contagion risk and the stochastic discount factor
L. Piccotti (2014)

Contagion risk (in the financial sector) can be estimated as follows:

- Start with bank returns, r_{it} ;
- Remove common risk factors (*e.g.*, Fama-French, estimated with a Kalman filter, assuming that the betas are random walks), and only look at the residuals e_{it} ;
- Regress those residuals against the sector residuals, with the bank of interest removed, $e_{it} = \beta_{it} e_{it} + \varepsilon_{it}$;
- Decompose the variance of the sector excess returns $\text{Var} \sum_i w_i e_{it}$ into a contagion contribution (the β_{it} , *i.e.*, the off-diagonal elements of $\text{Var } e_{\cdot t}$) and a residual term.

Explosive bubble modelling by noncausal process
C. Gouriéroux and J.M. Zakoian (2013)

Instead of a causal linear autoregressive process

$$y_{t+1} = \rho y_t + \varepsilon_{t+1}, \quad \varepsilon_t \text{ iid}$$

one can try to model financial time series with a *non-causal* AR process

$$y_t = \rho y_{t+1} + \varepsilon_{t+1}, \quad \varepsilon_t \text{ iid, Cauchy.}$$

[An AR(1) model is not convincing: try to add terms for 1-day, 1-week, 1-month, 1-year, *etc.*]

Stochastic areas of diffusions and applications in risk theory
Z. Cui

The area swept by a stochastic process below some threshold can be used to model default.

Extending time-changed Lévy asset models through multivariate subordinators
E. Luciano and P. Semeraro (2007)

To construct a multivariate Lévy process from a Gaussian one, use a multivariate subordinator, not a univariate one.

FARVaR
C.X. Cai et al. (2014)

To compute the value-at-risk, estimate the intraday density of returns (with a kernel estimator) and model the evolution of this density, from day to day, as a *functional autoregressive* (FAR) process.

Risk-adjusted time series momentum
M. Dudler et al. (2014)

Define risk-adjusted momentum as $\sum \frac{\log\text{-returns}_t}{\text{volatility}_t}$.

A Market-based funding liquidity measure
Z. Chen and A. Lu (2014)

The BAB (betting against beta) factor is the spread between (leveraged) low-beta stocks and (deleveraged) high-beta stocks. Its performance (low beta anomaly) can be explained by the inability of many investors to leverage their portfolio. But the beta is not a measure of funding liquidity: instead, one can combine size, idiosyncratic volatility, institutional ownership and analyst coverage – the BAB premium is higher for high-margin stocks. Appendix A lists 14 funding liquidity proxies.

Corporate news releases and equity vesting
A. Edmans et al. (2014)

CEOs time the release of corporate news to match their vesting equity.

Does liquidity beta predict mutual-fund alpha?
X. Dong et al. (2014)

The outperformance of a fund with a liquidity beta is not due to the liquidity premium, but to more trading opportunities in liquid times.

Policy announcements in FX markets
P. Mueller et al (2014)

Interest rate is a good FX investment signal, but only on scheduled FOMC announcement days. This can be explained by an agent model, in which agents learn the announcement effects in a Bayesian fashion.

Alliances and return predictability
J. Cao (2014)

Use alliance partners in your pairs trading strategy.

Post-split drift and post-earnings announcement drift: one anomaly or two?
L. Fengfei (2014)

Both stock splits and earnings announcements have a lasting effect on future returns, but the two phenomena may be distinct:

- The effects are real;
- They have the same duration, but
- A trading strategy using both outperforms one using only one.

[The conclusion is incorrect: the fact that the trading strategy performs better could come from the lower noise in the combined signal.]

International funding illiquidity
A. Vedolin (2014)

Country-specific funding illiquidity can be measured by the difference between market bond prices and theoretical bond prices (using the Svensson, Nelson-Siegel, or cubic spline model): large discrepancies mean that

arbitrageurs could not profit from the arbitrage opportunity.

Can information be locked-up? Informed trading ahead of macro-news announcements
Y. Tang (2014)

US macroeconomic announcements are often pre-released to the media under embargo agreements (lockup). Evidence (abnormal returns, order flow imbalance, from high-frequency data on index futures and ETFs, in the 30 minutes preceding the announcement) suggests that, for the FOMC (Federal Open Market Committee), this embargo is routinely breached – other government agencies manage to enforce it, though.

General purpose technologies, international technology diffusion, and the cross section of stock returns
W. Yang (2014)

Patents can be classified as general-purpose (computers, telecom, electronics) or special-purpose (chemical, medical, mechanical, etc.) The difference between the growth rate of general- and special-purpose patents is a non-diversifiable risk factors, that can be added to factor models (Fama-French, etc.)

Macro disagreement and the cross-section of stock returns
W. Li (2014)

High-beta stocks are more prone to speculative mispricing, and high dispersion (in market forecasts) leads to lower returns. This effect also exists for macroeconomic factors (industrial growth, inflation, etc.)

Liquidity risk and expected stock returns
L. Pástor and R.F. Stambauch (2003)

Stock liquidity can be defined as the volume coefficient in the regression

$$\text{return}_{t+1} \sim \text{return}_t + \text{sign}(\text{return}_t)\text{volume}_t.$$

The market liquidity is the equal-weighted average of the stock liquidities.

In search of distress risk
J.Y. Campbell et al. (2008)

Yet another bankruptcy model. [Why not a survival model? They already look at various horizons and the distance to default from Merton's model.]

The world price of political uncertainty
J. Brogaard et al. (2014)

Global political uncertainty (data from www.policy-uncertainty.com) affects the implied cost of capital of countries integrated with the global economy; local political uncertainty has the opposite effect (it can be defined as a binary variable indicating elections in a given

year – data from the World Bank database of political institutions). One can also use text data to measure political uncertainty:

- Proportion of articles about economic uncertainty;
- Residual of the regression of the domestic proportion against the global one.

A leverage-based measure of financial instability

A. Tepper and K.J. Borowiecki (2014)

Credit constraints, margin requirements and leveraged investors (their demand curve can become upward-sloping – to estimate it, assume they are optimistic and reinvest everything they gain, as much as possible) can make markets unstable. The stability condition (for a 4-agent model: leveraged investors, unleveraged investors, bank, central bank) can be written explicitly.

Aggregate short interest and return predictability
D.E. Rapach et al. (2014)

Detrended aggregate short interest helps predict future market returns.

Predictive systems under economic constraints
M. Bonelli et al. (2014)

Expected returns are not constant and can be modeled as

$$\begin{aligned} r_{t+1} &= \mu_t + u_{t+1} \\ x_{t+1} &= (1 - \lambda)x_\infty + \lambda x_t + w_{t+1} \\ \mu_{t+1} &= (1 - \beta)\mu_\infty + \beta\mu_t + v_{t+1} \\ \Sigma &= \text{Var}(u_t, v_t, w_t)'. \end{aligned}$$

Since the expected returns μ_t are unlikely to be negative, one can add this constraint, either with a Bayesian prior, or by reparametrizing the model, e.g., by using a square root process for μ_t .

Model uncertainty and expected return proxies
C. Jäkel (2013)

The implied cost of capital (internal rate of return in a dividend discount model – if you know the future dividends) is often used as a proxy for expected returns, but it is as imprecise.

Predicting the spread of financial innovations: an epidemiologic approach
I. Hull (2013)

The spread of financial innovations (new products: CDS, MBS, ABS, CDO, ETF, etc.) can be modeled with the SIR model. A 2-host SIR model (banks and creditors) can help monitor solvency problems.

Risk vs anomaly: a new methodology applied to accruals
J. Ohlson and P. Bilinski

Low accruals increase/decrease the probability of large positive/negative returns. (The “new methodology” is logistic regression...)

Strategic allocation to commodity factor premiums
D. Blitz and W. de Groot (2013)

Consider tercile commodity portfolios, for the following factors:

- Momentum (past 12-month return);
- Carry (annualized nearby future price / next nearby future price);
- Volatility (of daily returns, over the past 3 years)

The cross-sectional variation of volatility risk premia
A. González-Urteaga and G. Rubio (2014)

Instead of the index VRP (realized vol, minus model-free implied vol (MFIV)), look at the stock-specific one, and $\beta(\text{VRP}_{\text{stock}} \sim \text{VRP}_{\text{index}})$.

Convenient liquidity measure for financial markets
O. Danyliv et al.

There are already many definitions of liquidity: bid-ask spread, $|\text{returns}|/\text{ADV}$, long-term return variance / short-term return variance,

$$\frac{\text{high} - \text{low}}{\text{low}} \bigg/ \frac{\text{volume}}{\text{MCap}}.$$

Here is one more:

$$\log \frac{\text{Volume} \times \text{Close}}{\text{High} - \text{Low}}.$$

Colog asset pricing, evidence from emerging markets
Y. Dranev and S. Fomkina

Yet another measure of risk:

$$\text{Colog } X = E[X \log X] - E[X]E[\log X],$$

where X measures the loss beyond some threshold. One can also consider the colog utility:

$$U(x) = x - \lambda x \log x.$$

***Machines vs machines: high-frequency trading
and hard information***
Y. Huh (2014)

Hawkes's self-exciting model can be used to measure liquidity from high-frequency data (*i.e.*, how fast the intensity of the Poisson process reverts to its long-term value).

High-frequency trading both takes and provides liquidity, but the balance depends on information asymmetry – measure it by the proportion of transactions in the stock (say, AAPL) caused by a change in some index (sat QQQ).

Predictive power of aggregate short interest
E.J. Yu (2014)

Aggregate short interest can help predict cyclical GDP changes, up to four quarters ahead.

***Employee satisfaction, labor market flexibility,
and stock returns around the world***
A. Edmans et al. (2014)

Employee satisfaction only leads to higher returns in countries with a flexible job market (US, UK, but not Germany).

***Multi-scale representation
of high-frequency market liquidity***

A. Golub et al. (2014)

One can define another time scale (besides wall clock, volume clock, number-of-transactions clock) by looking at price changes beyond some threshold; time scales corresponding to different thresholds can be combined to define a multiscale market state, in $\{\text{up, down}\}^{\#\text{thresholds}}$.

***Predictability of bank stock returns
during the recent financial crisis***
W.S. Leung et al. (2014)

To predict the future returns of a bank, look at:

- Earnings;
- Non-performing loans;
- Loan-to-deposit ratio;
- Tier-1 capital ratio;
- Exposure to the structured finance market (off-balance sheet activities, ABX AAA index);
- Funding illiquidity risk (maturity mismatch).

The data for US banks comes from the bank regulatory database (“FRY5C form”).

Technical market indicators: an overview
J. Fang et al.

Technical analysis does not work – review of 93 indicators...

***Discretionary accrual models
and the accounting process***
X.G. Gómez et al. (2000)

Accruals should be included in the earnings: the cash flow alone is very noisy (its autocorrelation is negative), and accruals help smooth it. However, there is too much discretion in their computation: only part of them is informative.

Earnings = Cash flows + Good accruals + Bad accruals

The *non-discretionary accruals* have been defined as:

- Last year’s accruals;
- The average of the previous accruals;
- The fitted value of the following regression (divide by the previous assets and fit the models for each industry×year)

$$\text{Accruals} \sim \Delta(\text{Revenue} - \text{Receivables}) + \text{PPE}$$

$$\text{Accruals} \sim \Delta\text{Revenue} + \text{PPE} + \text{CFO}$$

$$\text{Accruals} \sim \text{Previous short-term accruals} + \text{Previous long-term accruals} + \Delta\text{CF}$$

where

$$\text{Short-term accruals} = \Delta((\text{Current assets} - \text{Cash}) - (\text{Current liabilities} - \text{Financing items}))$$

$$\begin{aligned} \text{Long-term accruals} = \\ - \text{Depreciation not in inventory} - \\ \Delta\text{Allowances. (?)} \end{aligned}$$

***Foreign ownership and real earnings
management: evidence from Japan***
J. Guo et al. (2014)

After the tightening of accounting rules in Japan (“big bang accounting reform”, in the late 1990s), companies have moved from accrual earnings management to real earnings management. This can be measured with the residuals of the following regression (divide by the previous assets and estimate for each industry×year pair, if there are at least 15 companies)

$$\begin{aligned} \text{Cash flow from operations} \sim \\ \text{Previous assets} + \text{Sales} + \Delta\text{Sales} \\ \text{Discretionary expenses} \sim \text{Previous assets} + \text{Sales} \\ \text{Production costs} \sim \\ \text{Previous assets} + \text{Sales} + \Delta\text{Sales} + \\ \text{previous } \Delta\text{Sales} \end{aligned}$$

where

$$\begin{aligned} \text{Discretionary expenses} &= \text{Advertising} + \text{R\&D} + \text{SGA} \\ \text{Production costs} &= \text{COGS} + \text{Inventory.} \end{aligned}$$

One can also add them:

$$\begin{aligned} \text{abnormal CFO} + \text{abnormal discretionary expenses} - \\ \text{abnormal production costs.} \end{aligned}$$

Firms with more foreign ownership are less manipulative.

***Performance matched
discretionary accrual measures***
S.P. Kothari et al. (2001)

Discretionary accrual measures, e.g., the residuals of the following regressions (divide by the previous assets, fit for each year×industry combination)

$$\text{Accruals} \sim \Delta\text{Sales} + \text{PPE}$$

$$\text{Accruals} \sim \Delta(\text{Sales} - \text{Receivables}) + \text{PPE}$$

$$\text{Accruals} \sim \Delta\text{Sales} + \text{PPE} + \text{Previous ROA}$$

$$\text{Accruals} - \text{Depreciation} \sim \Delta\text{Sales,}$$

rarely account for the relation between performance and accruals (a trend in performance leads to systematically positive/negative accruals). One can use *matching* (on year, industry, and previous ROA).

Kinetic component analysis
M. López de Prado and R. Rebonato (2014)

The authors have rediscovered structural models.

$$\text{acceleration} \quad a_{n+1} = a_n + \text{noise}$$

$$\text{velocity} \quad v_{n+1} = v_n + a_n + \text{noise}$$

$$\text{true price} \quad p_{n+1} = p_n + v_n + \frac{1}{2}a_n + \text{noise}$$

$$\text{observed price} \quad q_n = p_n + \text{noise}$$

(I think there is no $\frac{1}{2}a_n$ term in structural models, and R’s **StructTS** function only uses a first-order Taylor expansion, *i.e.*, assumes that the acceleration is zero.)

Those models can be fitted with a Kalman filter and, by choosing the amplitude of the noise, one can fine-tune the amount of smoothing.

Contrary to other smoothing methods, such as Fourier transform, wavelets, Hodrick-Prescott filter or local regression (lowess), which do not behave well at the ends of the sample, they are forward-looking, and provide forecasts and confidence intervals. For instance, market makers could use those forecasts to compute the acceptable interval for their quotes.

By looking if the acceleration is significantly different from zero, one can decide whether to follow a “momentum” strategy (strictly speaking, it should be called “inertia” strategy).

As usual, the article includes Python code.

***Reward-risk momentum strategies using
classical tempered stable distribution***
J. Choi et al. (2014)

Momentum strategies look at past performance but, instead of returns, one can consider other performance measures:

- VaR, CVaR, here estimated from and ARMA(1,1)-GARCH(1,1) model with classical tempered stable (CTS) innovations;
- Sharpe ratio, μ/σ ;
- Stable tail-adjusted return ratio (STARR), μ/CVaR , or an additive version of it, $\mu - \text{CVaR}$;
- Rachev ratio $\text{CVaR}(\text{gain})/\text{CVaR}(\text{loss})$.

Maximum drawdown, recovery and momentum
J. Choi (2014)

The returns over the past 6 months (or 6 weeks) can be decomposed into the sum of pre-peak, maximum drawdown, and recovery log-returns. In a momentum strategy, one can replace the past returns with various linear combinations of those three terms.

Persistent doubt: an examination of the performance of hedge funds
M. de la O González et al. (2014)

Hedge fund performance seems to be persistent: to check it, form quintile portfolios on various performance measures, such as

- Alpha, adjusted for everything you can think of (they used developed and emerging markets, size, bonds, credit, FX, commodities);
- The manipulation-proof performance measure (MPPM), $\Theta(A)$, *i.e.*, the certainty-equivalent monthly return (of the past 2 years of returns) for an investor with constant relative risk aversion (CRRA) $A = 3$;
- The excess MPPM, $\Theta_{\text{fund}}(A) - \Theta_{\text{Russell 2000}}(A)$;
- The doubt ratio, $\text{DR} = 2 + \Theta(2)/(\Theta(2) + \Theta(3))$.

Asymetric risks of momentum strategies
V. Dobrynskaya (2014)

The momentum anomaly can be explained by the CAPM with upside and downside betas: past winners have a lower β_+ and a higher β_- .

Bayesian analysis of bubbles in asset prices
A. Fulop and J. Yu (2014)

Bubbles can be described by regime-switching models (choose two among mean-reverting, unit root (random walk) and explosive root). Structural break models are similar, but their breakpoint is deterministic.

The authors suggest to model the price/dividend ratio (rather than the price) as a mean-reverting process, whose long-term mean follows a random walk, with an AR coefficient > 1 or < 1 depending on the regime, estimated with a particle filter.

emcee: the MCMC hammer
D. Foreman-Mackey et al. (2013)

The **Goodman-Weare sampler** uses the detailed balance rule, but the candidate distribution is inspired by differential evolution (DE):

- Consider n chains, X_1, \dots, X_n ;
- The candidate for X_k is $X_j + Z(X_k - X_j)$, where $Z \sim g$ is random, e.g., $g(z) \propto z^{-1/2} \mathbf{1}_{[a^{-1}, a]}(z)$ and $j \sim U(\llbracket 1, n \rrbracket \setminus \{k\})$.

The algorithm can be parallelized (but this changes the detailed balance condition) and is affine-invariant: it is not sensitive to different scales in the parameters.

emcee is a Python implementation.

GraphX: a resilient distributed graph system on Spark
R.S. Xin et al. (2013)

In a distributed environment, graph algorithms tend to be inefficient because data partitioning is often done on *edge cuts* (vertices are assigned to machines, edges span machines – it is a vertex-centric view of the graph). does not preserve locality. But graph partitioning with (random) *vertex cuts* (edges are assigned to machines, vertices span multiple machines – an edge-centric view) fares better.

GraphX (part of Spark) can easily implement the Pregel API (send-combine) or the Power Graph API (gather-apply-scatter).

Extreme learning machines: a survey
G.B. Huang et al. (2011)

“Extreme learning machines” are single-hidden-layer feed-forward neural networks in which the hidden layer is fixed and arbitrary – it seems to be another name for *echo state networks*, which can be seen as discrete support vector machines (with a random, rather than universal, kernel).

Variable selection using random forests
R. Genuer et al. (2012)

To account for groups of highly correlated explanatory variables, consider *nested random forests*, with an increasingly large number of variables.

Learning the parts of objects by non-negative matrix factorization
D.D. Lee and H.S. Seung (Nature, 1999)

Principal component analysis (PCA), vector quantization (VQ), nonnegative matrix factorization (NMF), independent component analysis (ICA) are all decomposition $V \approx WH$, with different constraints: orthogonal columns (rows) in H (W) for PCA; boolean W with exactly one 1 in each row for VQ; $W, H \geq 0$ for NMF. When used to reduce the dimension of a set of images (say, faces), they give qualitatively different results: PCA gives eigenfaces (not faces), designed to be combined; VQ gives prototypes (faces); NMF gives parts of images (the non-negative constraints only allow for additive reconstruction). On text data, the decompositions define topics: VQ only allows a single topic per text; PCA allows several, but these are “relative” topics, designed to be combined, and difficult to interpret; the independence assumption in ICA is inappropriate in this context.

When does the nonnegative matrix factorization give the correct decomposition into parts?
D. Donoho and V. Stodden

The nonnegative matrix factorization (NMF) $X = A\Psi$ can be interpreted geometrically: the rows of X are in the simplicial cone generated by the rows of Ψ , and this cone is in the first orthant. In particular, unless the cone is generated by rows of X “touches” the frontier of the orthant (e.g., the orthant itself, or an ice-cream cone tangent to the orthant), the decomposition is not unique: one can replace Ψ with a slightly wider cone. Separability and complete factorial sampling ensure unicity.

**Algorithms
for nonnegative matrix factorization
D.D. Lee and H.S. Seung**

A nonnegative matrix factorization (NMF) of an $n \times m$ matrix V (with nonnegative entries) is a decomposition $V \approx WH$ where W is $n \times k$ and H $k \times m$, with k small, and W and H have non-negative entries. By iterating

$$\begin{aligned} H &\leftarrow H \odot (W'V) \oslash (W'WH) \\ W &\leftarrow W \odot (VH') \oslash (WHH') \end{aligned}$$

(where \odot and \oslash are the elementwise multiplication and division), one can find a local minimum of $\|V - WH\|_2$ (it is actually a gradient descent with an adaptive learning rate). One can find a similar update rule to minimize the divergence $D(V \| WH)$, where

$$D(A \| B) = \sum_{ij} (A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij}).$$

Alternating least squares (ALS) were not mentioned.

**Big data in asset management
T. Roncalli (2014)**

Machine learning methods can be applied to finance:

- Lasso regression can be used for hedge fund replication, or to hedge positions;
- Nonnegative matrix factorization (NMF) can be used to decompose returns into factors, with positive weights (this is not the classical NMF: we want a decomposition $R = WF$, with $W \geq 0$, but no sign constraint on R or F ; no algorithm is mentioned, but alternating least squares (ALS) should work);
- There is no satisfactory *sparse Kalman filter*, but the model

$$\begin{aligned} y_t &= x_t' \beta_t + \varepsilon_t \\ \beta_t &= \beta_{t-1} + \eta_t \end{aligned}$$

can be estimated with a penalty, as in the Hodrick-Prescott (HP) filter,

$$\begin{aligned} (\hat{\beta}_1, \dots, \hat{\beta}_n) &= \underset{\beta}{\operatorname{Argmin}} \sum_t \|y_t - x_t' \beta_t\|^2 + \\ &\quad \lambda \sum_{t,j} h(\beta_{tj} - \beta_{t-1,j}) + \\ &\quad \mu \sum_{t,j} h(\beta_{tj}) \end{aligned}$$

with $h(\varepsilon) = \|\varepsilon\|_1$ or $h(\varepsilon) = \|\varepsilon\|_2^2$.

**Finding large average submatrices
in high dimensional data
A.A. Shabalin (2009)**

Biclustering algorithms look for homogeneous submatrices (e.g., approximately equal rows, or columns, or elements, or rank statistics in rows, or in columns, or elements above a threshold, or well-fit by a 2-way anova, etc.) and can be seen as a form of *unsupervised exploratory data analysis*. The LAS model assumes that the data is constant on (possibly overlapping) submatrices), with additive noise. The log-likelihood of a $k \times \ell$ submatrix of the whole $m \times n$ matrix is

$$-\log \left[\binom{m}{k} \binom{n}{\ell} \Phi(-\text{average} \times \sqrt{\ell}) \right]$$

To account for overlap, the submatrix with the best score is subtracted from the data, and the algorithm is iterated. To find a good submatrix with prescribed size $k \times \ell$, take k rows at random, take the best ℓ columns, then the best k rows, and iterate until convergence; then fine-tune (k, ℓ) .

**Measures of causality in complex datasets
with applications to financial data
A. Zaremba and T. Aste (2014)**

Granger causality is a comparison of $P(X_t | X_{\leq t-1})$ and $P(X_t | X_{\leq t-1}, Y_{\leq t-k})$. For a VAR model, one can use *Geweke's measure*,

$$F_{Y \rightarrow X} = \log \frac{\operatorname{Var}[X_t | X_{\leq t-1}]}{\operatorname{Var}[X_t | X_{\leq t-1}, Y_{\leq t-k}]},$$

or apply the kernel trick to it.

The *transfer entropy* is

$$T_{Y \rightarrow X} = H(X_t | X_{\leq t-1}) - H(X_t | X_{\leq t-1}, Y_{\leq t-k})$$

where

$$H(U|V) = \sum_{uv} p_{uv} \log \frac{p_v}{p_{uv}}$$

is Shannon's conditional entropy.

**Are there bubbles in stock prices?
Testing for fundamental shocks
A. Velinov and W. Chen (2014)**

A VAR(p) model, for a stationary time series y_t (if it is not stationary, try with Δy_t),

$$y_t = \nu + A_1 y_{t-1} + \dots + A_p y_{t-p} + u_t,$$

can be written

$$A(L)y_t = \nu + u_t$$

where L is the lag operator and (if $A(L)$ is invertible) as an MA(∞) process (Wold decomposition)

$$y_t = \mu + \sum_{s \geq 0} \Phi_s u_{t-s} = \mu + \Phi(L)u_t.$$

While the VAR parametrization is well-defined, the MA one is not identified: one can multiply u_t by an invertible matrix:

$$y_t = \mu + \Phi(L)B\varepsilon_t = \mu + \Psi(L)\varepsilon_t.$$

The article uses two variables,

$$y_t = \Delta \begin{pmatrix} \log(\text{production}) \\ \log(\text{stock prices}) \end{pmatrix}$$

and (to identify the model) the restrictions

$$E[\varepsilon_t \varepsilon_t'] = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \Psi = \begin{pmatrix} * & 0 \\ * & * \end{pmatrix}$$

(this is a **structural VAR** (SVAR) model). The two types of shock can be interpreted as fundamental and non-fundamental, and give a decomposition of the log-price into fundamental and non-fundamental components.

Downside volatility timing
I. Nolte and Q. Xu (2014)

Add (high-frequency, realized) downside risk to your alpha model or your risk forecasts (e.g., a heterogeneous autoregressive (HAR) model, $\sigma_{t,t+1} \sim \sigma_{t-1,t}^+ + \sigma_{t-1,t}^- + \sigma_{t-4,t} + \sigma_{t-22,t}$): it is more informative than volatility.

You can also try to decompose the risk into its jump and diffusion components (with the bipower variation).

***The economic value of realized jumps:
an asset allocation perspective***
I. Nolte and Q. Xu (2014)

Separating jumps from diffusion gives better volatility forecasts; they can be used in portfolio construction (“volatility timing”), either in the variance matrix in an optimization problem, or in a “parametric portfolio construction” (the portfolio weights are a linear function of some state variable, linked to the volatility components).

Volatility indices and state-preference pricing
Z.F. Liu (2014)

The VIX was originally computed from ATM options on the SP100 and the Black-Scholes formula. It is now computed from OTM SP500 options, but, although it no longer relies on the Black-Scholes formula, it is not completely model-free, ignores trading volume and includes illiquid options.

The article suggests a model-free approach (by constructing a “variance” option, that pays the squared log-returns) using only liquid vanilla options.

***Effects of financial crises on the long memory
volatility dependency of foreign exchange
rates: the Asian crisis vs the global crisis***
Y.W. Han (2014)

Long-memory, *i.e.*, the slow decay of the autocorrelation (processes between stationary and integrated) can be estimated with ARFIMA models (more precisely, for volatility, ARMA-FIGARCH) or by the local Whittle estimator (the asymptotic behaviour of the spectrum (periodogram) when $\omega \rightarrow 0^+$).

Omega risk model with tax
Z. Cui (2014)

Ruin theory models, such as the first hitting time

$$\tau = \inf\{t \geq 0 : X_t \leq 0\},$$

the deterministic grace period (Chapter 11)

$$\tau = \inf\{t \geq 0 : \forall s \in [t - \varepsilon, t] X_s \leq 0\}$$

or the stochastic grace period (omega risk model, $w(X_s)\mathbf{1}_{X_s \leq 0}$ is the intensity of a Poisson process)

$$\tau = \inf\{t \geq 0 : \int_0^t w(X_s)\mathbf{1}_{X_s \leq 0} ds > e\}$$

can be augmented with taxes (paid at rate γ when $X_t = \bar{X}_t$).

X_t = pre-tax value

U_t = after-tax value

$dU_t = dX_t - \gamma \bar{X}_t d\bar{X}_t$

\bar{X}_t = running maximum

***On a new index of riskiness:
theoretical results and some applications***
R. Resta and M.E. Marina (2013)

The riskiness of a gamble g can be measured by finding $\alpha = R_{AS}(g)$ such that $E[e^{-g/\alpha}] = 1$; this is monotonic wrt stochastic dominance. This can be generalized to $\alpha = I_\theta(g)$ such that

$$\frac{E[e^{-g-\cdot/\alpha} - 1]}{E[1 - e^{-g+\cdot/\alpha}]} = \theta.$$

Model risk of risk models
J. Danielsson et al. (2014)

To measure model risk, use several risk models (MA, EWMA, GARCH, t-GARCH, EVT, historical, etc.) and look at the ratio

$$\frac{\text{highest VaR (or ES)}}{\text{lowest}}.$$

The co-expected-shortfall is

$$\text{CoES} = E[X|M \leq M_\alpha],$$

where X are the stock returns, M the market returns, M_α the α -quantile of the market returns. The CoVaR is similarly defined.

On the measurement of economic tail risk
S. Kou and X. Peng (2014)

Median shortfall is the only “economically sound” (notion defined with a set of axioms) tail risk measure (yes, it is just a VaR).

The main problem (“elicitability”) with the expected shortfall (ES) is that, at least theoretically, you cannot check that your ES estimate is correct: something very bad could happen much farther in the tail, that would have no effect on the VaR, but a huge effect on the ES.

Power law and evolutionary trends in stock markets
P.V.S. Balakrishnan et al. (2007)

The power law exponent of stock trading volume has been increasing since 1965, suggesting a higher concentration of trades.

Rollover risk and volatility risk in credit spread models: a unified approach
S. Perrakis and R. Zhong (2013)

Credit risk is often estimated as a barrier option on the value of the firm, modeled as a random walk. The model can be improved by adding:

- The capital structure;
- Illiquidity risk – an illiquid bond market increases the *rollover risk*;
- Volatility risk – volatility is not constant (the article uses a CEV process).

Estimating stochastic volatility models using realized measures
J. Bekierman and B. Gribisch (2014)

Add the realized volatility to your GARCH or stochastic volatility model:

$$\begin{aligned}\text{return}_t &= e^{\lambda_t/2} \varepsilon_t \\ \log(\text{realized volatility}_t) &= \xi + \lambda_t + \sigma_1 u_t \\ \text{latent volatility} &= \lambda_t = \gamma + \delta \lambda_t + \sigma_2 \eta_t \quad ??? \\ \varepsilon_t, u_t, \eta_t &\sim N(0, 1).\end{aligned}$$

A user’s guide to the Cornish-Fisher expansion
D. Maillard (2012)

The Cornish-Fisher expansion is often considered as a black box: “compute the skewness and kurtosis of your distribution, plug them into the formulas, and retrieve the VaR and CVaR”. The expansion is actually very simple: a Gaussian random variable $Z \sim N(0, 1)$ can be transformed into a non-Gaussian one:

$$X = f(Z) = Z + \frac{S}{6}(Z^2 - 1) + \frac{K}{24}(Z^3 - 3Z) - \frac{S^2}{36}(2Z^2 - 5Z)$$

(understanding where this specific degree-3 polynomial comes from is less simple).

There are two common mistakes.

- To easily compute the VaR and CVaR of X from those of Z , f should be bijective: this imposes restrictions on the S and K parameters.
- The moments of X are not $(0, 1, S, 3 + K)$, but only asymptotically so, when $S, K \rightarrow 0$. In particular, what needs to be put in the formula is not the skewness and kurtosis, but the skewness and kurtosis *parameters* (the article fails to stress that the variance is affected by the same problem). The actual moments can be numerically transformed into the needed parameters.

Forward-looking measures of higher-order dependencies with an application to portfolio selection
F. Brinkmann et al. (2014)

Option prices can be used to estimate higher moments (skewness, kurtosis). In a multi-asset situation, the variance, skewness and kurtosis tensors can be modeled in a parsimonious way by assuming that the “higher correlations” (normalized covariance, coskewness, cokurtosis) are all equal, *i.e.*, besides the univariate parameters, there are only three parameters to estimate (ρ^{cov} , ρ^{coskew} , ρ^{cokurt}). Those higher moments can be used in portfolio optimization (take a 4th order Taylor expansion of the expected utility).

FVA explained: is there a solution?
D. Lu and F. Juan

When marking to market derivatives, one should also consider

- CVA (credit value adjustment), *i.e.*, the credit risk of the counterparty (CDS spread);
- DVA (debit value adjustment), *i.e.*, the credit risk of the bank itself;
- FVA (funding value adjustment), *i.e.*, the cost of funding the collateral for the derivative – uncollateralized derivatives are becoming rarer and rarer and the funding cost is not exactly the CDS spread.

Recovering from derivatives funding: a consistent approach to DVA, FVA and hedging
C.J. Gunnesson and A.F. Muñoz de Morales (2014)

CVA, DVA, FVA can be priced by replication.

KVA: capital valuation adjustment
A. Green et al. (2014)

Besides CVA, DVA, FVA, one can also include the cost of regulatory capital (KVA) in derivative valuation.

On multicurve models for the term structure
L. Morino and W.J. Runggaldier (2014)

To price forward rate agreements (FRA) and other fixed income products, you need a *multicurve model*,

with one curve for discounting, and one for the cash flows. [It had always shocked me to see people use the same curve.]

***Equity portfolio management
using option price information***
P. Christoffersen and X.N. Pan (2014)

Clear review of the use of options data for equity portfolio management:

- Smile slope = $\text{ivol}|K/S = 0.95 - \text{ivol}|K/S \approx 1$;
- $\langle \text{ivol} | \text{Call} - \text{ivol} | \text{Put} \rangle$;
- Model-free volatility, skewness, kurtosis;
- Exposure to VIX innovations, oil VIX (OVX), gold VIX (compute it yourself), copper VIX;
- Exposure to market skew;
- Exposure to VRP;
- Option-implied covariance matrix or, even, option-implied joint return distribution.

Dispersion trading in South Africa
S. Maze (2012)

A *dispersion trade* is long an index call, and short constituent calls; it is profitable if the constituents of the index are more volatile than the index.

***Option and accounting information: empirical
evidence in stock and derivative markets***
C.J. García Martín et al.

When forecasting the post-earnings announcement drift (PEAD), use options data (options volume/stock volume, open interest, etc.) if available.

***The price impact of option hedging
and the anomalous weekly reversal***
T. Wang (2014)

Hedging by option writers impacts prices, even for liquid underliers, especially around expiry, creating weekly reversals.

Identifying the drivers of predicted beta
J. Wand and J. menchero (MSCI, 2014)

One can compute betas from a factor model

$$r \sim N(\mu_0, V)$$

$$V = \text{eve}' + \Delta$$

$$\beta_i = \beta(r_i, w'r) = \frac{\text{Cov}(r_i, w'r)}{\text{Var}(w'r)} = \frac{V_{i, \cdot} w}{w'Vw}$$

and decompose it into factor contributions. One can do the same thing for the cross-sectional dispersion of the β 's.

Generalized risk-based investing
E. Jurczenko et al. (2013)

Minimum variance, maximum diversity, risk parity, etc. are special cases of the following optimization

problem:

$$\begin{array}{ll} \text{Find} & w \\ \text{To minimize} & \text{dispersion}_i \frac{w_i^\gamma}{\sigma_i^\delta} \frac{\partial \sqrt{w'Vw}}{\partial w_i} \\ \text{Such that} & w'\mathbf{1} = 1 \end{array}$$

i.e., trying to make those asset contributions equal, if possible.

***Equal-risk bounding is better than risk parity
for portfolio selection***
F. Cesarone and F. Tardella (2014)

The equal risk bounding (ERB) portfolio is the minimum risk portfolio among the risk parity portfolios built on a subsets of the assets:

$$\begin{array}{ll} \text{Find} & w \\ \text{To minimize} & \text{Max}_i w_i (Vw)_i \\ \text{Such that} & w'\mathbf{1} = 1, w \geq 0; \end{array}$$

it can also be written

$$\begin{array}{ll} \text{Find} & w, \lambda \\ \text{To minimize} & \lambda \\ \text{Such that} & \forall i w_i (Vw)_i \leq \lambda \\ & w'\mathbf{1} = 1, w \geq 0. \end{array}$$

***Efficiently combining multiple sources of alpha
in portfolio construction***
J. Menchero and J.H. Lee (MSCI, 2014)

Given a risk (or alpha) factor, one can define several portfolios. Let w_i be the weight of asset i in the market portfolio, with $\sum w_i = 1$. Let β_{ik} be the exposure of stock i to risk factor k , normalized: $\sum_i w_i \beta_{ik} = 0$, $\sum_i w_i \beta_{ik}^2 = 1$.

- The *simple factor portfolio* for factor k has weights $w_i \beta_{ik}$. (Thanks to the normalization) the factor returns can also be obtained by (univariate, time series) regression: $X_i = \alpha_i + \beta_{ik} F_k^{\text{simple}} + \varepsilon_i$.
- The *pure factor portfolio* for factor k has exposure 1 to factor k and exposure 0 to the other factors; the weights are the rows of $\beta^+ \text{diag}(w)$, where β^+ is the pseudo inverse of β ($\beta^+ = (\beta'\beta)^{-1}\beta'$ if the columns of β are linearly independent – it is a little more complicated otherwise, e.g., if you have world, country and sector factors).
- The *minimum volatility portfolio* among those with unit exposure to factor k .

***Alpha-risk factor misalignment:
does it pose a problem?***
J.H. Lee et al. (MSCI, 2013)

Portfolio optimization tends to maximize the exposure to the *residual alpha* – the part of the alpha not accounted for by the risk model –, assuming that it bears no risk. Since it does contain risk, and less information than the alpha, one can add a penalty for the exposure to this residual alpha to the optimization problem – but, quite often, this adjustment is not needed.

Diversified minimum variance portfolios
G. Coqueret (2014)

Regularization path of the minimum variance portfolio with an L^2 constraint.

Trilateral foreign exchange exposure
T.J. O'Brian (2014)

To compute (and hedge) the FX exposure of a firm in a trilateral situation (e.g., the firm and the customers use different currencies, and the price is in a third currency; two firms, from different countries, compete in a third country; firm, factory and market are in three different countries), consider unilateral shocks (*i.e.*, shocks, up or down, in one currency, leaving the other two fixed).

Pseudo-mathematics and financial charlatanism: the effects of backtest overfitting on out-of-sample performance
D.H. Bailet et al. (Notices of the AMS, 2014)

The dangers (and omnipresence) of overfitting in finance.

Significance testing in empirical finance: a critical review and assessment
J.H. Kim and P.I. Ji (2014)

Empirical finance is still in the statistical dark ages: people use conventional (arbitrary) significance thresholds, disregarding:

- Sample size;
- Economic significance (often, tests do not answer the question “is there something” but “do I have enough data to see it?”): do not test $H_0 : \beta = 0$ against $H_1 : \beta > 0$, but against $H_1 : \beta > \beta_1$ – if you do not want to choose β_1 , report a confidence interval for β ;
- Test power;
- Heteroskedasticity, autocorrelation, outliers;
- Expected losses.

One could use:

- Smaller thresholds (10^{-3} is fine in psychology);
- Bayesian methods, *i.e.*, check if

$$\frac{P(H_1|\text{data})}{P(H_0|\text{data})} = \frac{P(\text{data}|H_1) P(H_1)}{P(\text{data}|H_0) P(H_0)} > 1.$$

In addition, the publication bias is high and replicated studies almost inexistent. [I disagree: I see a lot of replicated studies – but they remain unpublished.]

Adaptive learning and survey data
A. Markiewicz and A. Pick (2013)

Homo economicus and his exact subjective probabilities should be replaced with *homo econometricianus*, who uses time series models fitted on real data (“adaptive learning”) and macroeconomic data.

Quantifying differential interpretation of public information
X.S. Sheng

By modeling analysts as Bayesian learners, blending their prior beliefs with their interpretation of new information, one can decompose the dispersion in their forecasts into a prior dispersion and the dispersion due to new information,

$$\text{Var}_{\text{new information}} = \frac{\text{Var } X_i^{\text{before}} - \lambda^2 \text{Var } X_i^{\text{after}}}{1 - \lambda^2}$$

$$X_i^{\text{after}} - X^{\text{after}} = \alpha + \lambda(X_i^{\text{before}} - X^{\text{before}}) + \varepsilon$$

Investor behavior and financial innovation: a case study on callable bull/bear contracts
X. Li et al. (2014)

Knock-out barrier options are cheap, volatile, and skewed – prospect theory can explain why investors like them (as a lottery), even though they lose money.

Bankruptcy sells stocks... but who's buying and why?
L. Coelho et al.

Bankrupt companies are actively traded, and often overpriced: they are used by retail investors as lottery stocks (high probability of a small loss, low probability of a large gain, *i.e.*, high skewness).

Pairs trading with copulas
W. Xie et al. (2014)

When implementing a pairs trading strategy (once you have identified the pair, e.g., via cointegration tests), one often just looks at the difference in normalized prices,

$$p_X(0) = p_Y(0) = 1$$

$$\Delta(t) = p_X(t) - p_Y(t)$$

$$\text{Trade when } |\Delta(t)| > 2\sigma_{\Delta(t)}.$$

Instead, one can estimate the copula of the returns, and the conditional probabilities (“mispricing indices”)

$$\text{MI}_t^{X|Y} = P[X_t < x_t | Y_t = y_t]$$

$$\text{MI}_t^{Y|X} = P[Y_t < y_t | X_t = x_t]$$

(where X_t are the returns and x_t the realized returns) and trade when $\sum_{s \leq t} \text{MI}_s^{X|Y}$ or $\sum_{s \leq t} \text{MI}_s^{Y|X}$ exceeds some threshold.

Improving pairs trading
T.R. Almeida (2011)

Pairs opening after a 1-sided shock are less profitable.

***Liquidity-adjusted price-dividend ratios
and expected returns***
B.G. Jang et al.

The price-dividend ratio is often assumed to be stationary: it can therefore be defined as a cointegration relation for $(\log(\text{price}), \log(\text{dividends}))$. But evidence suggests that it may not always be stationary: one can try to replace it with a cointegration relation for

$$(\log(\text{price}), \log(\text{dividends}), \log(\text{liquidity})),$$

where the USD trading volume can be used as a proxy for liquidity.

Random walks in dividend yields and bubbles
F. Bidian (2014)

A non-stationary dividend yield is not necessarily evidence of a bubble.

Estimating private equity market beta using cash flows: a cross-sectional regression of fund-market paired internal rates of return
Y. Jiang and J. Sáenz (2014)

To estimate the beta of private equity (PE):

- Compute the internal rate of return (IRR) of the PE investments (you need several of them, and you get an IRR for each);
- Compute the corresponding market IRRs (using the same cash flows, but invested in the market, with a non-zero final value);
- Regress.

[I am skeptical about the stability of this procedure: the market IRRs are weighted averages of market returns – they will be very, very similar, so we are almost regressing against a constant...]

***Peering inside the analyst “black box”:
how do equity analysts model companies?***
A. Markou and S. Taylor (2014)

Analysts value firms using the discounted cashflow model (DCF), with explicit cashflow forecasts (subjective, or from subjective forecasts of balance sheet items, or from subjective/historical growth rates thereof). The discount factor (weighted average cost of capital, WACC) is computed from the cost of debt, the cost of equity (equity risk premium, ERP), the tax advantage of debt, and the capital structure – some include preference shares and pension liabilities in the capital structure. The ERP is difficult to estimate – most use the CAPM. The terminal value and the time after which it is reached influence the result. The DCF is complemented by other approaches: EV/EBITDA, P/E, etc.

For more details, check Damodaran’s books.

***Changes in cash:
persistence and pricing implications***
J.Z. Chen and P.B. Shane (2010)

The changes in earnings can be decomposed into

$$\Delta \text{Earnings} = \text{Accruals} + \Delta \text{FCF}$$

$$\Delta \text{FCF} = \Delta \text{Cash} + \text{net distribution to debt holders} \\ + \text{net distribution to shareholders.}$$

The change in cash can be decomposed into a normal and an abnormal part,

$$\Delta \text{Cash} = \widehat{\Delta \text{Cash}} + \text{residual}$$

where the forecast $\widehat{\Delta \text{Cash}}$ comes from the model

$$\Delta \text{Cash} \sim \sigma_{\text{industry}}(\text{FCF}) + \\ \Delta \frac{\text{assets} - \text{BV}(\text{equity}) + \text{MV}(\text{equity})}{\text{assets}} + \\ \Delta \log(\text{assets}) + \Delta \text{FCF} + \\ \frac{\text{working capital} - \text{cash}}{\text{assets}} + \\ \Delta \frac{\text{debt}}{\text{assets}} + \Delta \frac{\text{R\&D}}{\text{assets}} + (\Delta \text{dividends} > 0) + \\ \Delta \frac{\text{cash outflow on acquisitions}}{\text{assets}} + \\ \Delta \text{Cash}_{t-1} + \text{Cash}_{t-1}.$$

Negative abnormal cash changes are persistent, and do not bode well for the company. Positive abnormal cash changes are not persistent, but such hubris is ignored by the market.

***When the use of positive language backfires:
the joint effect of language sentiment,
readability and investor sophistication
on earnings judgements***
H.T. Tan et al. (2013)

When readability is low, sentiment (*i.e.*, word choice) matters, but in opposite directions for naive and sophisticated investors.

R&D spillover and predictable returns
Y. Jiang et al. (2012)

R&D spending is an externality:

- Low R&D firms lag high R&D firms in the same industry;
- The surprises are higher for low R&D firms (in the same industry) that are not often found next to the leaders (in news, portfolios or analysts’ reports).

***What do we learn from two new
accounting-based stock market anomalies?***
S. Basu (2004)

Accounting anomalies (buy low NOA, sell short firms receiving a going-concern audit opinion, etc.) are consistent with *minimally rational markets*: the prices are

inconsistent with all investors being rational, but transaction costs are too high for rational investors to make a profit.

The roles of receivables and deferred revenues in revenue management
J. Zha (2014)

To detect manipulations, look at:

- $\Delta \text{receivables} \gg 0$ (fictitious sales, bill-and-hold);
- $\Delta \text{deferred revenue} \ll 0$ (get the cash early, ship the goods late).

Disentangling the accruals mispricing in Europe: is it an industry effect?
E. Basilico and T. Johnsen (2013)

Also look at:

- $\frac{\Delta \text{accounts receivables}}{\text{net operating assets}};$
 $\frac{\Delta \text{inventory}}{\text{net operating assets}};$
- Days of sales outstanding = $\frac{\Delta \frac{\text{receivables}}{\text{sales}}}{\frac{\text{sales}}{\text{receivables}}};$
- Days of inventory outstanding = $\frac{\Delta \frac{\text{inventory}}{\text{COGS}}}{\frac{\text{inventory}}{\text{COGS}}}.$

R in Finance 2014

Packages. Among the packages presented, beyond the usual `data.table` and `Rcpp` (with C++11 features), and the omnipresence of `shiny`:

- The next edition of *Modeling Financial Time Series with SPlus* ditches SPlus (and will contain a list of must-know R packages);
- `eventstudies` for the Patell test;
- `FlexBayes` fits hierarchical models (linear, logistic, Poisson, via MCMC);
- `ilmts` will provide Hurst exponent estimation and simulation of long (and intermediate) memory processes;
- `cds` implements the ISDA standard model for CDSs, *i.e.*, the standardized characteristics of the contract and the quoting conventions;
- The `pbo` package estimates the probability of back-test overfitting.

R engine. Recent or forthcoming performance improvements include:

- Bytecode compilation;
- Shallow duplication (since R 3.1.0);
- Reference counting (perhaps in R 3.2.0);
- Large vectors (> 16 GB);
- Parallelization: explicit parallelization works; C/C++ parallelization via OpenMP only works on Linux; implicit parallelization only works for a few functions;
- The `proftools` package will soon be available.

Portfolio construction. The `PortfolioAnalytics` package performs portfolio optimization (with ROI (`Rglpk`, `Rsymphony`, `quadprog`), random portfolios, `DEoptim`, `pso`, `GenSA`), for various types of objectives (mean, value at risk (VaR), expected shortfall (ES), standard deviation, expected utility) and constraints; the types of objective and constraints are apparently limited.

The `cccp` package will solve cone-constrained convex programs.

One should include taxes (in particular, “tax harvesting” effects in case of losses) in portfolio optimization: it can be very profitable to replace a stock with an equivalent one.

Let \mathbf{x} be the (random variable of) asset returns and $\tilde{\mathbf{x}} = (1 \ \mathbf{x}')'$. The inverse of

$$\Theta = E[\tilde{\mathbf{x}}\tilde{\mathbf{x}}'] = \begin{pmatrix} 1 & \mu' \\ \mu & \Sigma + \mu\mu' \end{pmatrix}$$

is

$$\Theta^{-1} = \begin{pmatrix} 1 + \mu'\Sigma\mu & -\mu'\Sigma^{-1} \\ -\Sigma^{-1}\mu & \Sigma^{-1} \end{pmatrix} = \begin{pmatrix} 1 + \text{sharpe}^2 & -\mathbf{w}' \\ -\mathbf{w} & \Sigma^{-1} \end{pmatrix}$$

where \mathbf{w} is the markowitz portfolio. The distribution of $\text{vech } \Theta^{-1}$ is asymptotically gaussian, and $\text{Var } \text{vech } \Theta^{-1}$ is easy to compute (from $\text{Var } \text{vech } \Theta$): it can be used to perform tests or compute confidence intervals on the Shape ratio or the portfolio weights. This can be generalized to constrained portfolios (e.g., the optimal portfolio with zero covariance wrt some reference portfolio).

To check if a signal has some predictive power on future returns, build the top and bottom quintiles, and match them (on industry, size, trading volume): if the effect disappears, it was caused by the imbalance between the portfolios. Coarsened exact matching is implemented in the `cem` package.

Cointegrated pairs do not remain so (use the `egcm` package for the tests).

Higher moments. `DEoptim` can account for higher moments in portfolio optimization.

The factor model $r_i = \sum_k b_{ik}f_k + e_i$ (with $E[r_i] = E[f_k] = E[e_i] = 0$, $f_k \perp e_i$, $e_i \perp e_j$ if $i \neq j$) can be used to decompose the higher moments of the portfolio returns.

$$E[(w'r)^2] = \sum_{ijkl} w_i w_j b_{ik} b_{jl} E[f_k f_l] + \sum_i w_i^2 E[e_i^2]$$

$$E[(w'r)^3] = \sum_{\substack{i_1, i_2, i_3 \\ k_1, k_2, k_3}} w_{i_1} w_{i_2} w_{i_3} b_{i_1 k_1} b_{i_2 k_2} b_{i_3 k_3} E[f_{k_1} f_{k_2} f_{k_3}] + \sum_i w_i^3 E[e_i^3]$$

$$E[(w'r)^4] = \dots$$

(I leave the fourth moment as an exercise to the reader: contrary to the second and third moments, the cross-products do not all disappear – you should have three

more terms.) Those higher-order tensors can be written as matrices using the Kronecker product

$$\begin{aligned} E[(w'r)^2] &= w'(BSB' + \Delta)w \\ E[(w'r)^3] &= w'(BG(B' \otimes B') + \Omega)(w \otimes w) \\ E[(w'r)^4] &= w'(BP(B' \otimes B' \otimes B') + Y)(w \otimes w \otimes w). \end{aligned}$$

Volatility models. The 2-state STAR model

$$y_t = [\lambda_t \phi_1 + (1 - \lambda_t)]' \begin{pmatrix} 1 \\ y_{t-1} \\ \vdots \\ y_{t-p} \end{pmatrix} + \varepsilon_t$$

smoothly switches between two AR models, ϕ_1 and ϕ_2 ; the state transition function $\lambda_t = \lambda(z_t)$ depends (via a logistic link) on exogenous variables z_t . It can be extended to a STARMAX model by allowing autoregressive dynamics in the state z_t .

One can use the bootstrap to estimate the precision of volatility indices (VIX, vega-weighted VIX (VVIX), liquidity- or elasticity-weighted VIX); check the `ifrogs` package for an implementation.

The `stochvol` package uses MCMC to fit stochastic volatility (SV) models: for time series modeling, regression with SV noise or factor models.

The `gpustcalibration` package uses GPUs to calibrate stochastic volatility models; it relies on `nloptr` and `DEoptim`.

Risk models. The `factorAnalytics` package can estimate risk factor models (time series, fundamental, statistical) and decompose the risk (standard deviation, VaR, ES) into risk factor contributions;

To estimate correlation, one can use shrinkage or, even better, weighted averages of shrinkage estimators.

When monitoring the volatility (or the VaR) of a portfolio, $\text{risk} = \sqrt{w'Vw}$ where $V = \beta v \beta' + \Delta$, one may want to decompose its changes into contributions of changes in weights w , changes in β , changes in v , changes in Δ . Graphically, one can look at:

$$\begin{aligned} \text{risk}(\text{portfolio}_{\text{now}}, \text{model}_t) &\sim t \\ \text{risk}(\text{portfolio}_t, \text{model}_{\text{now}}) &\sim t \\ \text{VIX}_t &\sim t \\ \text{VaR}_t^{99\%} - \text{VaR}_t^{95\%} &\sim t \end{aligned}$$

For stress-testing:

- Model the joint distribution of the risk factors and the stress factors (the stress factors may be in your risk model), either as a Gaussian or a mixture of Gaussians;
- Sample from the conditional distribution

$$\text{risk factors} \mid \text{stress factors}$$

for given values of the stress factors (−10%, −20%, −30%, etc.)

One can add the volume synchronized probability of informed trading (VPIN) to the Fama-French, momentum and liquidity factors; look at the beta of stock VPIN vs Market VPIN.

Performance measurement. To measure the performance of private equity (PE), use the cash flows to and from the fund, apply them to the benchmark, and look at $\log_{10}(\text{IRR}_{\text{PE}}) - \log_{10}(\text{IRR}_{\text{Market}})$ or

$$\frac{\text{final value}_{\text{PE}}}{\text{final value}_{\text{Market}}}.$$

Time series. To predict economic recessions, one can:

- Take high-frequency time series (e.g., daily index returns);
- Compute their spectrograms on a moving window (short-term Fourier transform (STFT), sometimes also called Gabor transform); one can notice more low-frequencies in expansion periods, and more mid-frequencies in recessions;

```
library(quantmod)
getSymbols( "^GSPC", from="1980-01-01" )
x <- log1p( as.vector( ROC( Ad( GSPC ) ) ) )
dates <- index(GSPC)
```

```
library( e1071 )
y <- stft(x, inc=1)$values
dates <- index(GSPC)
dates <- tail( dates, nrow(y) )
```

```
par(mar=c(2,1,1,1), xaxs="i")
image(
  dates, 1:ncol(y), y,
  col = rev(topo.colors(10)),
  axes = FALSE, xlab = "", ylab = ""
)
axis.Date( 1, dates )
box()
par(new = TRUE)
plot(
  dates, rowSums(y),
  type = "l", lwd = 3, axes = FALSE
)
```

```
# The patterns we see are just due
# to the volatility: they completely
# disappear if we normalize the signal
# in each window...
```

```
image(
  dates, 1:ncol(y), y/rowSums(y),
  col = rev(topo.colors(10)),
  axes = FALSE, xlab = "", ylab = ""
)
axis.Date( 1, dates )
box()
par( new = TRUE )
plot(
  dates, rowSums(y),
  type = "l", lwd = 3, axes = FALSE
)
```

- Consider those spectrograms as images, and model them as a 2-dimensional Gaussian process;
- Compute the *empirical orthogonal functions* (EOF) of the sequence of images (basically, a PCA): this gives variables with a potential predictive power on the current or future market state;
- Add low-frequency variables (GDP, perhaps also yield, etc.);
- Use *stochastic search variable selection* (SSVS) on those variables to find a parsimonious probit model, to predict the current or future state of the economy.

The `bcp` package detects change points (different means) in multivariate (Gaussian) time series, using bayesian methods. Use with log-returns (I would also try with their absolute value) for a few indices or currency pairs.

Tests. `robust::lsRobTest` provides tests on robust models (I often say that you should start to worry and pay attention to robust methods (e.g., `robust::lmRob`) when they yield significantly different results).

Text mining. When extracting sentiment from text, the position of the words is relevant: for instance, if a text starts with a lot of negative words and ends with a lot of positive words, it does not mean it is neutral. Use weighted measures of sentiment, with $\frac{1}{2}, \frac{1}{3}, \frac{2}{3}$ (1, x , $x(1-x)$), etc. as weights (they like $P_c(u) - (1-u^c)u^{3-c}$).

Networks. From the adjacency matrix A of a graph, one can compute the number of paths of length m between i and j as $[A^m]_{ij}$. The total number of paths between i and j , $[\sum_{m \geq 0} A^m]_{ij}$, may be infinite: instead, one can give less importance to longer paths, $[\sum_{m \geq 0} w_m A^m]_{ij}$, e.g., $[\exp A]_{ij}$. Many graph notions (centrality, communicability, etc.), usually computed from the adjacency matrix A , can be generalized and computed from $f(A)$ (with, e.g., $f = \exp$): the f -centrality $e'_i f(A) \mathbf{1}$ generalizes the degree, the f -communicability $f(A)_{ij}$ generalizes the number of paths. Check the `irlba` package (implicitly-restricted Lanczos method) to help compute those quantities on large, sparse matrices.

Infrastructure. The `RHIPE` package is an interface to Hadoop; `datadr` provides a simpler divide-map-reduce API, using either Hadoop (via `RHIPE`) or and in-memory key-value store. `treilliscope` applies the same ideas to treillis/lattice/faceted plots; it relies on “cognostics” (e.g., scagnostics, i.e., scatterplot cognostics) to navigate the huge number of panels.

The `rredis` package serializes/deserializes R objects to/from strings with `paste/substitute/deparsed`: it is slow for large objects (e.g., time series). `RapiSerialize` does the serialization/deserialization in C++; `RcppRedis` uses it, and can therefore efficiently deal with streaming data: for instance, C++ or Python would put the data to process in Redis, and R would read and process it.

OneTick is a tick data database, with online aggregation and complex event processing (CEP) in a GUI; the

node can be arbitrary R/Python/Java/C++ code: for instance, one could fit a state-space model (well, PCA) on a moving window to predict future returns.

CEM: Software for coarsened exact matching S.M.Iacus et al. (2009)

Coarsened Exact Matching (CEM) removes the sampling bias in observational studies: it temporarily coarsens each control variable (i.e., bins the observations and builds a histogram) and prunes the observations in the (multivariate) bins that do not have at least one control and one treated unit. The statistical analysis can then proceed on the remaining data (with weights inversely proportional to the bin sizes).

Asset Allocation with Higher Order Moments and Factor Models K. Boudt et al (2014)

An approach for identifying and predicting economic recessions in real-time using time-frequency functional models S.H. Holan et al. (2012)

Apparent criticality and calibration issues in the Hawkes self-excited point process model: application to high-frequency financial data V. Filimonov and D. Sornette (2013)

Hawkes processes (Poisson processes in which each event increases the background Poisson intensity, with an exponential or power law decay – there are several commonly-used such “kernels”) can be interpreted as branching processes (each event is either a mother event, exogenous, coming from the background intensity, or a daughter event, endogenous, coming from a previous event). The *branching ratio* is the average number of daughter events.

Goodness of fit can be assessed with a change of time, $\hat{\Lambda} = \int \hat{\lambda}$, which should give a Poisson process with $\lambda \equiv 1$.

The calibration of Hawkes processes is plagued by the following problems:

- Sensitivity to outliers (use a contamination model);
- Edge effects: missing mother events before the sample, missing daughter events after;
- Sensitivity to kernel misspecification: we often only care about the tail of the kernel, but its short-term behaviour also matters;
- Multiple extrema in the log-likelihood.

The microstructure of high-frequency data brings more problems:

- Overnight trading is not negligible, and it is different;
- The exchange only sends 1-second timestamps, and groups the ticks (to send fewer messages); The data provided tries to add miliseconds using the reception

time (ignoring latencies – problematic if they fluctuate), but there are still ties (one can try to add noise to reduce the bias);

- The data is not stationary, $\mu(t)$ is not constant: there are daily patterns (U shape) (estimate the pattern and remove it to get $\lambda \equiv 1$ – but this distorts the kernel) and announcement days are completely different (remove them);
- Regime switches (between different branching ratios, or different background intensities) are interpreted as excess clustering and bias the estimation.

An investigation of trades that move the BBO
Y. Huang et al.

The bid-ask spread is insufficient to measure liquidity: one can look into the depth of the order book by considering *strings*, *i.e.*, series of trades with non-increasing prices and computing the (average) number of trades, total volume, duration, log-return, volatility, beginning spread, beginning trade size, beginning price.

To test for variations between days or securities, one can use the intra-class correlation coefficient (ICC).

Investor networks in the stock market
R. Bilkdik et al. (2013)

While it is possible to develop an information network model and estimate it using (approximate) maximum likelihood from transaction data, this is not scalable. In a very empirical way, one can estimate the network by linking agents i and j if they trade the same asset, in the same direction, in a window of size τ , at least M times.

Agents with higher centrality trade earlier and earn higher returns.

A longitudinal analysis of asset return, volatility and corporate news network
G. Creamer et al. (2012)

Use a topic model (LDA, latent Dirichlet analysis) on a newsfeed, build a bipartite network of topics and companies (mentioned in the news items), reduce it to a network of companies (with the number of articles on the same topic as weights) and test for Granger causality between graph metrics (centrality, etc.) and returns or volatility.

A link mining algorithm for earnings forecast and trading
G. Creamer and S. Stolfo (2008)

Build a bipartite graph with directors (or directors and analysts) and firms as nodes, reduce it to a graph of firms (with the number of directors or analysts in common as weights): the network metrics (centralities, clustering coefficient) can help predict earnings surprises.

Trading networks
L. Adamic et al. (2010)

A simple limit-order book model (n brokers, a long liquidity taker, a short liquidity taker) suggests that order flow imbalance and size are visible on the network. Using transaction-level data, one can build a time-varying transaction network (e.g., on a 600-transaction window), look at network metrics

- Centrality (in-degree – out-degree);
- Absolute centrality;
- Assortativity (degree correlation);
- Clustering coefficient (comparison of the number of triangles and the number of connected triples);
- Size of the largest strongly connected component

and financial variables

- Amihud price impact;
- Bid-ask spread;
- Volatility (absolute value of the returns, squared returns or high – low);
- Intertrade duration;
- Trading volume;

Network metrics Granger-cause volume and intertrade duration.

The two faces of interbank correlation
K. Schaeck et al. (2013)

The correlation between banks can be decomposed into a diversification component (if the banks are very diversified, they are doing all possible banking activities, *i.e.*, they are all doing the same things – they are therefore exposed to the same sources of risk) and a residual component (containing, among others, the endogenic risk resulting from the relations between the banks).

$$\text{Commonality}_A = \text{Cor}(\text{returns}_A, \text{returns}_{\text{sector}})$$

$$\text{Diversification}_A = \text{Cor}(\text{returns}_A, \text{returns}_{\text{market}})$$

$$\text{Excess commonality}_A =$$

$$\text{residuals}(\text{Commonality} \sim \text{Diversification})_A.$$

If one knew the bank holdings, one could define commonality (resp. diversification) as the distance between the weights of the portfolio of bank A and the aggregate portfolio of the banking sector (resp., the market portfolio), and the excess commonality as the difference between the commonality and the minimum commonality possible for that level of diversification.

Filtering noise from correlation matrices
A. Izmailov and B. Shay (2013)

Random matrix theory (RMT) is often used to clean sample correlation matrices: since the asymptotic ($T \rightarrow \infty$, $N \rightarrow \infty$, $Q = T/N$ constant) distribution of the eigenvalues of the sample correlation matrix of a standard Gaussian random variable is known and bounded, one can discard all eigenvalues beyond the (asymptotic) maximum

$$\lambda_{\max} = (1 + Q^{-1/2})^2.$$

The article suggests a finite sample correlation (but this increases λ_{\max} , which is already very high) and measures the noise present in the correlation matrix by the “entropy”

$$\sum_i \frac{\lambda_i}{n} \log_2 \frac{\lambda_i}{n}.$$

Since, on the examples, the effect of filtering seems to be an increase in the “contrast” of the correlation matrix, one may want to compare with a simple coefficient-wise transformation, e.g., $x \mapsto x^\alpha$.

Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms

S. Salvador and P. Chan

To estimate the number of clusters, consider some performance metric as a function of the number of clusters, and try to find the “knee” in its plot: $\text{Argmax}_i x_{i+1} - x_i$, $\text{Argmax}_i x_{i+1}/x_i$, $\text{Min}\{i : D^2(x)_i = x_{i+1} + x_{i-1} - 2x_i \geq c\}$, $\text{Argmax}_i D^2(x)_i$, $\text{Argmax}_i \text{res}(x \sim i)$, or the changepoint of a broken line fit to the data. If there is a lot of data, use the method again on $\llbracket 1, 2 \rrbracket$ and iterate until convergence.

The relative asset pricing model: towards a unified theory of asset pricing

A. Muraalidhar et al.

Advocacy for liability-driven investing (LDI): institutional investors (pension funds form a growing proportion of all investors) first define a “liability proxy portfolio”, describing their future liabilities (most do not do that seriously and settle for “60% equity, 40% bonds”) and then try to track or beat it. This can be modeled in a CAPM-like way,

$$E[X - X_0] = \beta E[M - L - X_0],$$

where X are the asset returns, X_0 are the risk-free returns, M are the market returns, L are the liability portfolio returns. The model can be augmented with more factors, à la Fama-French, and the presence of a reference point L accounts for prospect theory.

A robust capital asset pricing model

D. Ruffino (2014)

In mean-variance portfolio optimization

$$\hat{w} = \text{Argmax}_w \text{CE}[w'X] \approx \text{Argmax}_w w'\mu - \frac{1}{2}\lambda_1 w'V_1 w,$$

only the returns $X \sim N(\mu, V_1)$ are uncertain. In robust mean-variance optimization, μ and V_1 are also uncertain. It is possible to linearize the certainly equivalent into

$$\hat{w} \approx \text{Argmax}_w w'\mu_0 - \frac{1}{2}\lambda_1 w'V_1 w - \frac{1}{2}\lambda_2 w'V_2 w$$

where $\mu \sim N(\mu_0, V_2)$. (Note that the uncertainty on V_1 does not appear in this approximation.)

This is equivalent to the initial problem with V_1 replaced with $V_1 + (\lambda_2/\lambda_1)V_2$ – i.e., this looks like a shrinkage estimator.

The Fama-French three factors in the Chinese stock market

J. Xu and S. Zhang (2013)

The Fama-French model works well in China (higher returns for small or high-value stocks, higher R^2 when compared with the US), but pay attention to a few pitfalls:

- A large proportion of the shares are (still?) non-tradable;
- There are 4 markets (main Shanghai board, main Shenzhen board, SME, GEB – the last two, also in Shenzhen, are similar to the Nasdaq: you may, or not, want to include them);
- There are multiple share classes (A, B, H, foreign) – do not forget them when computing per-share quantities.

The determinants of convenience yields

M. Prokpczuk and Y. Wu (2013)

The carry cost is the difference between the future and spot prices. It corresponds to interest foregone, storage costs, inventory, insurance against unexpected demand, insurance against supply shocks, speculation, etc. – it can be interpreted as a convenience yield.

Rather than the spread future – spot, the convenience yield can be estimated from a model: the spot is a diffusion, the convenience yield is an Ornstein-Uhlenbeck process and it determines the spot trend). The convenience yield can be explained by:

- Global inventory levels;
- Hedging pressure (relative net/gross position for commercial traders);
- Volatility;
- Macroeconomic conditions: industrial production and inflation forecasts (from the BCEI).

Systematic tail risk

M.R.C. van Oordt (2013)

The tail beta, the sensitivity to extreme (say, beyond the 5% value at risk) market returns is persistent and a good predictor of losses during crises. However, there is no tail beta premium.

Patent- and innovation-driven performance in venture capital-backed IPOs

J. Cao et al (2013)

Patents (use the NBER patent database) are still a good sign.

***General purpose technologies
and the cross-section of stock returns***
P.H. Hsu and W. Yang

Classify patents as general-purpose (computers, telecom, electronics) or special-purpose (chemical, medical, mechanical, etc.). The difference between the growth rate of general and special purpose patents predicts industrial growth; it is a non-diversifiable risk factor, that can be added to your factor risk models (Fama-French, etc.).

***Blockholder exit threats
and financial reporting quality***
Y. Dou (2014)

The presence of blockholders (measured by the Herfindahl index of shareholders above 5%), increases financial reporting quality (measured by abnormal accruals, production costs, discretionary expenses, operating cash flow, where “abnormal” means “absolute value of the residuals of a regression against other financial variables, cross-sectionally, by industry), at least for liquid stocks.

***Environmental disclosure
and the cost of capital: evidence
from the Fukushima nuclear accident***
P. Bonetti et al. (2013)

Environmental disclosure (CO₂ emissions, etc.) is associated to a lower cost of capital (and a lower rise in cost of capital after accidents). It can be measured by: the existence of an environmental report, the inclusion of CO₂ emissions, the inclusion of a CO₂ emissions target, lower CO₂ emissions than the industry. The effect on the cost of capital can be estimated with *propensity score matching* wrt size, leverage, ROA, B/P industry.

Institutional presence
J. Sulaeman and C. Wei (2013)

Non-shareholder institutional observers improve corporate governance (and liquidity, cost of capital, information diffusion). It can be measured by the assets under management of institutional investors located in the region (US state) where the firm is headquartered.

Property theft and the cost of capital
J.D. Brushwood et al. (2013)

High state-level property crime leads to a higher cost of capital.

***Linguistic complexity in firm disclosures:
obfuscation or information?***
B.J. Bushee et al. (2013)

Linguistic complexity (measured by some linear combination of the average number of words per sentence, the average number of syllables per word, the proportion of “complex” words: Gunning fog index, Fleisch-Kincaid index) in manager-driven discourse is a sign

of obfuscation, but it is informative in analyst-driven discourse.

One can assess the relation with *information asymmetry*, measured by

- Amihud liquidity = $\frac{|\text{daily returns}|}{\text{dollar volume}}$;
- $\lambda_{GH} = 2(\lambda_1 + \lambda_2 \times \text{average size})$, where

$$\Delta p_t = \psi_1 \Delta D_t + \psi_2 (\Delta D_t \cdot \text{Size}) + \lambda_1 D_t + \lambda_2 D_1 \cdot \text{Size} + \text{noise}$$

p_t = price

D_t = sign(trade_t);

- λ_{MRR} in

$$\frac{\Delta p_t}{p_t} = \psi \Delta D_t + \lambda (D_t - \rho D_{t-1}) + \text{noise}$$

$$D_t = \text{noise} + \rho D_{t-1}.$$

Liquidity risk in credit default swap markets
B. Junge and A.B. Trolle (2013)

Market illiquidity can be measured by the difference between the price of an index CDS and its theoretical price, from the CDSes of its constituents. The corresponding index arbitrage portfolio is a *tradable liquidity factor*. It can be used in a multi-factor model, as we do for equities.

***Managing option trading risk with Greeks
when mental accounting matters***
H. Siddiqi

Many people consider call options as a surrogate for the underlying – they end up with biased Greeks and one can profit from them.

***Breaking bitcoin: does cryptocurrency
exchange activity lead to increased real activity
outside cryptocurrency exchanges?***
D. Vitt (2013)

One can use Bitcoin’s *perfect ledger* of all transactions to model the evolution of volume, price, number of transactions (with a distinction between hot (casinos) and cold (real economy) wallets), search frequency (Google trend), money supply with a VAR model and impulse-response plots.

***When finance meets physics: the impact
of the speed of light on financial markets
and their regulation***
J.J. Angel (2014)

The regulators have forgotten that information does not travel instantaneously (e.g., there are 4ms between Chicago and New York): rules such as “best execution” are actually unimplementable. Clock synchronization difficulties also prevents them from reconstructing what happened.

***ParadisEO-MOEO: a framework
for evolutionary multi-objective optimization***
A. Liefooghe et al. (2007)

ParadisEO is a C++ framework for multiobjective optimization, *i.e.*, for the estimation of the Pareto frontier. Here are some of the ideas used in those algorithms:

- Pareto ranking: a solution's rank is the number of solutions dominating it (rank-1 solutions are optimal, at least in the sample);
- Pareto front: one can group the candidate solutions into “fronts” of increasing Pareto rank, to preserve diversity;
- One can keep both an archive of Pareto optimal solutions and a current population; the strength of an archive solution is the number of population solutions it dominates; the strength of a population solution is the sum of the strengths of the archive solutions dominating it.

***The area under the ROC curve
and its competitors***
J. Hilden (1991)

There are a few variants of the ROC curve:

- OROC (ordinary ROC),

$$P(\text{positive}|D) \sim P(\text{positive}|\neg D);$$

- FROC (frequency-scaled ROC),

$$P(\text{positive}, D) \sim P(\text{positive}, \neg D);$$

- EUROC (expected utility ROC),

$$E[U(\text{treatment})|x, D] \sim E[U(\text{treatment})|x, \neg D]$$

The AUC can be interpreted as the probability of correctly distinguishing between a diseased (D) and a non-diseased ($\neg D$) patient, but it is not helpful for a single patient. The power of a diagnostic test (x) is better measured by its *diagnosticity*:

$$\text{Prior regret} = -\text{Max}\{E[U(T)], E[U(\neg T)]\}$$

$$\text{Posterior regret} = -\text{Max}\{E[U(T)|x], E[U(\neg T)|x]\}$$

$$\text{Diagnosticity} = \text{Prior regret} - E[\text{Posterior regret}]$$

***Use and misuse of the receiver operating
characteristic curve in risk prediction***
N.R. Cook (2007)

Do not use the AUC (or its generalization for survival data, the c statistic) for model selection

***Visualizing distributions
of covariance matrices***
T. Tokuda et al.

To visualize a distribution of covariance matrices (e.g., a prior), look at:

- Univariate distributions of the log-volatilities and the correlations (since distributions used as priors are often exchangeable, there are few of them to consider: $\log \sigma_1$ and ρ_{12});
- Bivariate scatterplots;
- Trivariate scatterplot of correlations

$$\begin{pmatrix} 1 & \rho_{12} & \rho_{13} \\ \rho_{12} & 1 & \rho_{23} \\ \rho_{13} & \rho_{23} & 1 \end{pmatrix};$$

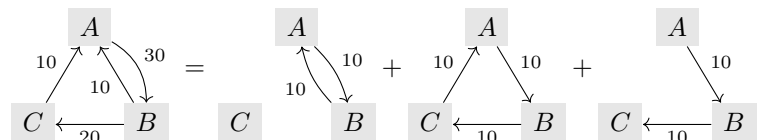
- A hundred 50% equiprobability matrices for $N(0, \text{Var}(X_1, X_j))$;
- Histogram of the *effective variance* $|V|^{1/n}$ and the *effective dependence* $1 - |\text{cov2cor}(V)|^{1/n}$ (highlight samples with a high or low effective dependence in the other plots);
- The effective dependence of $V_{1:k,1:k}$ versus k .

Commonly-used priors are not as uninformative as we would like: for instance, for the inverse Wishart, ρ_{ij} is uniform, but (ρ_{ij}, ρ_{kl}) is not; even for a uniform correlation, the effective dependence depends on the size and converges to $1 - e^{-1}$.

Those plots are implemented in the **VisCov** R package.

***Trade interpretation and trade imbalances
in the European union: a network perspective***
G.M. Krings et al. (2013)

Flow, in a trade network, can be decomposed into symmetric, cyclic (often negligible) and acyclic components.



There are usually several acyclic flow graphs: choose the most diverse (maximize the sum of squared flows – Herfindahl index, *i.e.*, a generalized entropy). The acyclic graph can be decomposed into (single-source, single-destination) elementary flows: start with the nodes with no inflows as sources and no outflows as destinations (as in a topological sort), compute the maximum flow (not unique if there is a bottleneck), and choose the most diverse one.

Integration between two countries can be defined as trade/GDP.

One can also consider the random walk of a dollar on the trade network (PageRank without teleportation): the average commute time can be used as a distance; after multi-dimensional scaling (MDS), one can watch the time evolution of the network.

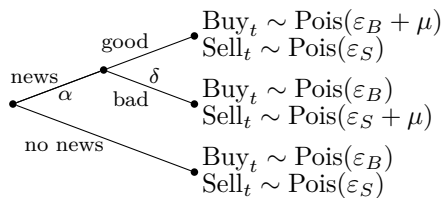
Political risk spreads
G. Bekaert (2013)

The political risk spread is the proportion of the sovereign spread explained by the *political risk rating* from the ICRG (international country risk grade), in a regression that also includes global macroeconomic conditions (high-yield spread), local macroeconomic conditions (GDP, GDP growth, inflation, budget, current account) and liquidity (incidence of zero daily bond returns).

One can also look at the components of the political risk rating.

Cluster PIN: a new estimation method for the probability of informed trading
Q. Gan et al. (2013)

The PIN (probability of informed trading) model



$$PIN = \frac{\alpha\mu}{\alpha\mu + \varepsilon_B + \varepsilon_S}$$

is difficult to fit. One can approximate it by considering the order imbalance $X_t = \text{Buy}_t - \text{Sell}_t$; its distribution is a mixture of three *Skellam distributions* (differences of independent Poisson variables), corresponding to the three regimes (no news, bad news, good news). It is not easier to fit, but one can use hierarchical clustering to approximate the three regimes; the resulting parameters can then be used as starting points for a more exact algorithm.

Short-horizon CEO incentives and abnormal stock returns, insider trading and earnings management
J.D. Chi et al. (2013)

Firms with short-horizon CEO incentives (data from S&P ExecuComp) and high short sales constraints (*i.e.*, with inefficient information dispersal, as measured by idiosyncratic volatility, short interest and size) are different: share price inflation and correction, greater earnings surprises, greater responses to earnings surprises, higher accruals, higher revenue for the CEO selling his shares.

To control for exogenous variables, the study uses *propensity score matching*: first, predict the probability that a firm has short-horizon CEO incentives, with a logit model from: incentive ratio (stock and option compensation, divided by total compensation), P/B , total assets (size), ROA, volatility of cash flow from operations, capital intensity (fixed assets / total assets), governance (institutional ownership, Gompers's governance index, percentage of independent directors

on the board, board size); then, compare the long- and short-horizon CEO incentive firms with a similar predicted probability of short horizon.

Lowball guidance and management's credibility
J. Chen (2013)

IBES provide management compensation term structure data.

Economic linkages inferred from news stories and the predictability of stock returns
A. Scherbina and B. Schlusche (2013)

Stock co-occurrence, in news, is informative: price shocks (after a bankruptcy, an oil spill, etc.) diffuse on that graph, making 1-day returns predictable.

Asset volatility
M. Correia et al. (2013)

Structural bankruptcy models (Merton) can be improved by better estimates of asset volatility:

- $\sigma = \lambda \sigma_{\text{Market Value}}$, where $\lambda = MV/(MV + \text{debt})$;
- $\sigma = \sqrt{\lambda^2 \sigma_{MV}^2 + (1 - \lambda)^2 \sigma_{\text{debt}}^2 + 2\rho \sigma_{MV} \sigma_{\text{debt}}}$, where ρ is shrunk towards to industry average and σ_{debt} is estimated from a representative bond;
- Replace σ_{MV} with the implied volatility (average the put and call volatilities), which is more forward-looking;
- The volatility of the RNOA (return on net operating assets), estimated from the inter-quartile range, from the quantile regression

$$RNOA_{t+1} \sim RNOA_+ + (RNOA > 0) + RNOA_- + \text{accruals} + \text{dividends} + (\text{dividends} > 0)$$

- The volatility of the seasonality-adjusted RNOA, $RNOA_t - RNOA_{t-4Q}$.

Instead of a probit model

$$\text{bankruptcy} \sim \text{leverage} + \text{past returns} + \text{market capitalization} + \text{volatility},$$

a *survival model* (predicting the time to bankruptcy) makes better use of the data.

Buffet's alpha
A. Frazzini et al. (2013)

Berkshire Hathaway's returns (at least the public stock portfolio part – the private equity part and the value of Buffet himself are difficult to replicate) can be replicated using market, size, value, low-beta and quality factors, with a 1.6 leverage – Berkshire have a low financing rate, thanks to their AAA rating, the use of insurance as liabilities, deferred taxes due to accelerated depreciation (free loans from the IRS), and the sale of derivatives.

A five-factor asset pricing model
E.F. Fama and K.R. French (2013)

The Fama-French model to explain excess returns can be extended by adding two more factors:

- Market excess returns;
- Returns of the quintile long-short size portfolio;
- Returns of the quintile long-short B/P portfolio;
- Returns of the quintile long-short profitability portfolio, where

$$\text{profitability} = \frac{\text{revenue} - \text{COGS} - \text{interest} - \text{SGA}}{\text{book equity}};$$

- Returns of the quintile long-short investment portfolio, where

$$\text{investment} = \frac{\Delta \text{assets}}{\text{previous assets}}.$$

The model is not sufficient to describe returns: the intercept is significantly different from zero – however, it is small, so it may not matter after all.

***The other side of value:
the gross profitability premium***
R. Novy-Marx (2012)

Another article on gross profits/assets.

***In short supply:
equity overvaluation and short selling***
M.D. Beneish et al. (2013)

Short selling facilitates information flow, and makes markets more efficient. Short selling constraints, e.g., lack of supply of shares to borrow, impede information flow: if you use the short interest ratio (SIR), consider correcting it for insufficient supply; the cost of borrow can be used as a proxy for supply.

(No mention of the *diversity* of supply.)

***Risk-averse reinforcement learning
for algorithmic trading***
Y. Shen et al.

Reinforcement learning can be made risk-averse: just replace the expected future reward with its expected utility, for some concave utility function.

***The meaning and use of the area under a
receiver operating characteristic (ROC) curve***
J.A. Hanley and B.J. McNeil (1982)

The area under the ROC curve can be interpreted as the probability that a randomly chosen pair of a positive and a negative observation is correctly identified; it can also be seen as a Wilcoxon statistic, which leads to sample size and power computations.

***Generating multi-asset arbitrage-free
scenario trees with global optimization***
A. Consiglio et al. (2013)

When generating scenarios for a stochastic optimization problem, one has to ensure that they are arbitrage-free. Checking if they are is a convex problem, but finding arbitrage-free scenarios that match given moments is a non-convex problem. It can be solved efficiently via a sequence of convex relaxations.

***How high frequency trading
affects a market index***
D.Y. Kenett et al. (2013)

This is not what the article is about, but the abstract suggested a study of the *term structure of beta*.

***Can Google Trends search queries
contribute to risk diversification?***
L. Kristoufek (2013)

Search volume, from Google trends (use the ticker, perhaps prefixed with “stock”), may be a proxy for risk and can be used in portfolio construction

$$\text{weight} \propto \text{search volume}^\alpha.$$

There are no comparisons with other measures of risk, with other weighting schemes (market capitalization), and the study was limited to the Dow Jones 30.

***Stochastic dominance tests
on the ASEAN40 index***
V. Aumeboonsuke (2011)

One can use stochastic dominance to compare investments or asset returns: increase the order until the difference becomes significant.

When to sell a Markov chain asset?
Q. Zhang (2013)

Computation, using the Bellman equation, of the optimal stopping time to sell a stock whose log-price follows a telegraphic process (random motion with two possible slopes, with slope changes determined by an (observable) 2-state Markov chain – equivalently, the slope changes are determined by a Poisson process).

***The fine structure of volatility feedback II:
overnight and intra-day effects***
P. Blanc et al. (2013)

Intraday and overnight returns can be modeled jointly; they behave and interact differently.

GARCH and FIGARCH models are special cases of

ARCH(∞):

$$\begin{aligned}\text{ARCH}(q) : \quad \sigma_t^2 &= s^2 + \sum_{\tau=1}^q K(\tau) r_{t-\tau}^2 \\ \text{GARCH} : \quad \sigma_t^2 &= s^2 + \sum_{\tau \geq 1} g e^{\tau/\tau_p} r_{t-\tau}^2 \\ \text{FIGARCH} : \quad \sigma_t^2 &= s^2 + \sum_{\tau \geq 1} g \tau^{-\alpha} e^{\tau/\tau_p} r_{t-\tau}^2\end{aligned}$$

Statistical inference of comovements of stocks during a financial crisis
T. Ibuki et al. (2013)

The Ising model can be used to model buy (+1) and sell (−1) decisions; the returns are $P(+1) - P(-1)$. The energy is

$$E = -\alpha \sum_{ij} S_i S_j - \beta \sum_i R S_i - \gamma \sum_i M S_i,$$

where $S_{i,k}(t) \in \{\pm 1\}$ is the decision of trader i at time t for asset k , $R = \frac{1}{N} \sum_i S_i(t-1)$ are the previous returns, $M = \frac{1}{K-1} \sum_{\ell \neq k} c_{k\ell} m_\ell$, $c_{k\ell}$ is the correlation between assets k and ℓ , m_ℓ is the behaviour of the average trader for asset ℓ .

The parameters α, β, γ can be estimated; their evolution over time can highlight regime changes; they can be used to forecast returns (e.g., currencies).

copulaedas: and R package for estimation of distribution algorithms based on copulas
Y. Gonzalez-Fernandez and M. Soto (2013)

Estimation of distribution algorithms (EDA) is a population-based global optimization algorithm:

- Start with a set of candidates, uniformly sampled in the feasible region;
- Fit a distribution (Gaussian or, as in this article, based on a multivariate copula or a vine copula);
- Sample a few candidates from this distribution;
- Replace some of the old candidates;
- Iterate.

(You may want to improve the candidates with a local search as they are generated.)

Similar software include: ParadisEO (C++) and MathEDA (Matlab).

Insider trading in Hong Kong: concentrated ownership versus the legal environment
J. Zhu et al. (2002)

Looking at Jensen's alpha (the intercept in the CAPM model) or comparing firms with their peers (build $3 \times 3 \times 3$ tercile portfolios on size, B/P and momentum) suggests that insider purchases are informative, while their sales are not. In contrast with the US, the effect is more pronounced for large (not small) companies.

Portfolio concentration and the geometry of co-movement
W. Phoa (JPM, 2013)

Diffusion maps are another dimension reduction technique (the article fails to compare it with principal component analysis (PCA), multi-dimensional scaling (MDS), independent component analysis (ICA), isomap, t-SNE, Kohonen maps, etc.):

- Compute the similarity matrix $1 + \text{Cor}(X)$;
- Divide each row by its sum, to have a probability transition matrix;
- Compute its first k eigenvectors, rescale them by the eigenvalues, and use their coefficients as coordinates.

One can estimate the *portfolio concentration* by computing the variance matrix Σ of the resulting cloud of points (using the portfolio weights) and looking at $\sqrt{\text{tr } \Sigma}$. One can estimate the *local concentration* with a kernel estimator.

This can be generalized to other measures of dependence (rank correlation, tail dependence, etc.)

An introduction to diffusion maps
J. de la Porte et al. (2008)

For the purpose of dimension reduction, small distances are more informative than large distances. Since multi-dimensional scaling (MDS) weighs them in the same way, it struggles at identifying non-linear, low-dimensional structures. Isomap uses the geodesic distance on a graph, but is not very robust to perturbations. Instead, one can consider a diffusion process (a random walk – this is very similar to the PageRank algorithm) on the complete graph, with transition probabilities

$$p(x \rightarrow y) \propto \exp - \frac{\|x - y\|^2}{2\sigma^2}.$$

Diffusion maps are projections to a lower-dimensional space coming from the spectral analysis of the transition matrix P (or P^t , after t steps) – note that this matrix is not symmetric. Those projections approximate the *diffusion distance*

$$D_t(i, j) = \sum_k |P_{ki}^t - P_{kj}^t|^2$$

(two points are close if they are similarly distant from any other third point).

Here are the computations: let K be the kernel matrix, $D = \text{diag}(K\mathbf{1})$ the (normalizing) density, $P = D^{-1}K$ the transition matrix; $Q = D^{1/2}PD^{-1/2}$ is symmetric and can be diagonalized $Q = SAS'$; $P = (D^{-1/2}S)\Lambda(D^{-1/2}S)^{-1}$ is also diagonalizable and $D^{-1/2}S = [\psi_0|\psi_1|\dots|\psi_n]$ are its right eigenvectors; send observation i to $(\lambda_1^t \psi_1(i), \dots, \lambda_k^t \psi_k(i))$ for some $k \in \llbracket 1, n \rrbracket$ and $t > 0$.

What is... Data Mining
M. Maggioni (2012)

Gentler presentation of diffusion maps.

Speaker identification using diffusion maps
Y. Michalevsky et al. (2011)

To use diffusion maps for classification tasks (e.g., speaker identification), one needs to project *new* observations on the submanifold, e.g., using the *Nyström extension*.

$$k(x, y) = \exp - \frac{\|x - y\|^2}{2\sigma^2}$$

$$d(i) = \sum_j k(x_i, x_j)$$

$$p(i, j) = \frac{k(x_i, x_j)}{d(i)}$$

The right eigenvectors ψ_k satisfy $P\psi_k = \lambda_k\psi_k$, *i.e.*

$$\psi_k(i) = \frac{1}{\lambda_k} \sum_j p(i, j)\psi_k(j).$$

That formula can be generalized to project any point z , not just one of the x_i :

$$d(z) = \sum_j k(z, x_j)$$

$$p(z, x_j) = \frac{k(z, x_j)}{d(z)}$$

$$\psi_k(z) = \frac{1}{\lambda_k} \sum_j p(z, j)\psi_k(j).$$

[There may be a confusion, in this article or in others, between eigenvectors and singular vectors, and/or between the matrices $D^{-1}K$ and $D^{-1/2}KD^{-1/2}$.]

Diffusion maps
R.R. Coifman and S. Lafon (2006)

Original article on diffusion maps, in a continuous context. One can define a *family* of diffusions (a diffusion is a continuous analogue of a random walk) as

$$k_\sigma(x, y) = \exp - \frac{\|x - y\|^2}{2\sigma^2}$$

$$d_\sigma(x) = \int k_\sigma(x, y) dy$$

$$k_{\sigma,\alpha}(x, y) = \frac{k_\sigma(x, y)}{d_\sigma(x)^\alpha d_\sigma(y)^\alpha}$$

$$d_{\sigma,\alpha}(x) = \int k_{\sigma,\alpha}(x, y) dy$$

$$p_{\sigma,\alpha}(x, y) = \frac{k_{\sigma,\alpha}(x, y)}{d_{\sigma,\alpha}(x)},$$

with $\alpha = 0, \frac{1}{2}$ or 1 (there are three parameters: α , σ and t , the number of steps in the random walk). With $\alpha = 1$, the distribution of the points on the submanifold has no influence.

**Diffusion maps and coarse-graining:
a unified framework for dimensionality
reduction, graph partitioning
and data set parametrization**
S. Lafon and A.B. Lee (2006)

For large datasets, one can aggregate the data and control the discrepancy on the diffusion distances.

Diffusion maps for signal processing
R. Talmon et al. (2013)

Diffusion maps can be used in signal processing:

- For single-channel source localization, one can record white noise from various angles, compute the corresponding diffusion map, notice that its first component ψ_1 corresponds to the location (angle), and project any new recording onto it.
- To identify repeating, transient interference (keyboard noises, etc.), one can compute the diffusion map of the frames of the signal and notice that they form two clusters: a compact one, with the interference, and a more diffuse one, with the signal (speech). The interference samples can be averaged (the speech samples are too different: averaging them is destructive).
- Non-local filtering averages a signal x with nearby signals: $\hat{x} \propto \sum_i \bar{k}(x, x_i)x_i$; the kernel \bar{k} used can come from the diffusion distance:

$$\bar{k}(x, y) = \exp - \frac{D_t^2(x, y)}{2\varepsilon^2}.$$

To build diffusion maps, the Euclidian distance is not always the best choice. Consider:

- Euclidian distance in feature space;
- Intrinsic distance: cluster the data, compute the covariance C_i of each cluster i , use the Mahalanobis distance

$$d(x, y) = \frac{1}{2}(x - y)'(C_{\text{cl}(x)}^{-1} + C_{\text{cl}(y)}^{-1})(x - y).$$

- Local PCA models: use dimension reduction, but only locally, to remove the noise when computing small distances (they matter more than large distances).

The diffusion map can be computed from the singular value decomposition (not checked). Let K be the kernel matrix, $D = \text{diag}(K\mathbf{1})$ the (normalizing) density, $P = D^{-1}K$ the transition matrix, $P = U\Lambda V'$ its singular value decomposition, $V = [\psi_0|\psi_1|\dots|\psi_n]$ its right singular vectors, $\Lambda = \text{diag}(\lambda_0, \lambda_1, \dots, \lambda_n)$ its singular values; send observation i to $(\lambda_1^t\psi_1(i), \dots, \lambda_k^t\psi_k(i))$ for some $k \in \llbracket 1, n \rrbracket$ and $t > 0$.

Applications of machine learning to finance
Z. Cazalet and R.L. Dao

Lasso regression can be used to replicate hedge funds with a parsimonious portfolio.

The *non-negative matrix factorization* (NMF)

$$A = BC$$

with $\forall i, j, k \ A_{ij}, B_{ik}, C_{kj} \geq 0$, A square, B and C rectangular and small can be used as an alternative to principal component analysis (PCA) or independent component analysis (ICA), with log-prices (not price returns); stocks can be clustered using the corresponding factors.

When fitting classification models (probit, etc.), try ensemble methods such as *bagging* or *boosting*.

The *Hodrick-Prescott* (HP) filter is a low-pass filter with an L^2 penalty on the second derivative,

$$\underset{y}{\operatorname{Argmin}} \frac{1}{2} \|y - x\|_{L^2}^2 + \lambda \|D_2 x\|_{L^2}^2.$$

One can use an L^1 penalty instead: the L^2 penalty tends to “oversmooth” the signal and the L^1 penalty tolerates isolated singularities.

Add support vector machines (SVM), with cross-validation, to your toolbox. They can also be used to smooth a signal [not clearly explained].

Experiments in conditioning risk estimates with quantified news
C. Kantos (Northfield, 2013)

Risk models are backward-looking, but they can be adjusted with contemporaneous data. GARCH models are sometimes used for this, but they do not deal well with announcement days. One can use intraday volatility, cross-sectional volatility, implied volatility (VIX if the stocks does not have a liquid options market), variation in news sentiment. Only the factor and stock-specific variances are adjusted, not the exposures or the correlations.

[Data sources: Ravenpack, Alexandria, Market Psych, Thomson Reuters, Recorded Future (sentences in the future tense).]

Cointegration strategies for asset allocation
D. diBartolomeo (Northfield, 2013)

Cointegration can provide “active returns with passive management”: find a portfolio (or a continuously-rebalanced strategy) with price P such that $P - \text{Price}(\text{index} + 3\%)$ be stationary – for hedge fund returns, use $P - \text{Price}(10\%)$. This can be described as “liability-driven investing”. The results are more robust than modern portfolio theory. VAR (vector autoregressive) models are the other traditional way of incorporating non-contemporaneous structure in time series.

An innovative look at corporate credit risk
S. Malinak (Northfield, 2013)

There are many ways of estimating the probability of default:

- Structural models (Merton, KMV): the value of the assets of a company can be modeled as a geometric random walk (estimate its volatility from the volatility or implied volatility of the stock and the capital

structure; somehow estimate its drift) and one can compute the probability that the value of the assets will drop below that of the debt (Black-Scholes formula – barrier options seem to make more sense, but simpler vanilla options actually perform better);

- Accounting models (Altman, Ohlson, etc.);
- Forward-looking accounting models, with analyst estimates instead of past reported values (when available);
- Text mining from company filings, transcripts, news feeds or broker research (the data has to be cleaned of compliance and commercial verbiage, and a sentiment lexicon specific to that kind of document is needed).

Incorporating commodities in the multi-asset-class portfolio risk assessment
D. diBartolomeo (Northfield, 2013)

Adding commodities to a risk model is not as straightforward as it may seem:

- There are too many of them: one should use a factor model, e.g., by clustering commodities into agricultural, energy, precious metals, industrial metals, performing a principal component analysis (PCA) in each cluster, and retaining the first two components (as suggested by random matrix theory (RMT));
- The correlations (between commodities, between commodities and other asset classes) are very volatile (the confidence intervals are very large, and often contain 0), even after *GARCH correction*;
- There is a lot of overlap with other asset classes: instead of adding the commodity factors to the model, one can express them using the factors already there – since it works, it suggests that, except perhaps for agricultural commodities, commodities hardly add any diversification to a portfolio.

Liquidity planning tools and strategy capacity for equity markets
D. diBartolomeo (Northfield, 2013)

The risk of a portfolio increases with size. To estimate it, decide on a *liquidation policy* (cost of liquidating $x\%$ in n days in a normal/crisis situation), compute the volatility of the portfolio, convert it to a value at risk (VaR), add the liquidity costs (from your transaction cost model), and convert the corrected VaR back to a volatility.

Quality minus junk
C.S. Asness et al. (2013)

The discounted cash flow model

$$\text{Price} = \sum_{n \geq 0} (1+r)^{-n-1} d(1+g)^n = \dots = \frac{d}{r-g},$$

where d is the next dividend, $(1+r)^{-1}$ the discount

factor, and g the growth rate, can be written

$$\begin{aligned} P/B &= \frac{d/B}{r-g} \\ &= \frac{\frac{\text{Profit}}{\text{Book value}} \times \frac{\text{Dividends}}{\text{Profit}}}{r-g} \\ &= \frac{\text{Profitability} \times \text{Payout ratio}}{\text{Required return} - \text{Growth}}; \end{aligned}$$

this suggests to combine various measures of profitability, payout (shareholder friendliness), required return (safety) and growth, in the hope that the errors will average out. A quality indicator can be constructed by adding the z -scores (rank the stocks, divide the rank with the number of stocks, apply the Gaussian inverse cumulative distribution function) of various financial ratios falling in those categories:

- Profitability: GP/A (gross profit to assets), ROE (return on equity), ROA (return on assets), CFOA (cash flows over assets), gross margin, accruals;
- Growth: change in those factors over the past 5 years;
- Safety: Altman z -score, Ohlson model, beta, leverage, ROE volatility;
- Payout: new equity, new debt, payout-to-profits (dividends paid and shares repurchased, divided by sales minus COGS (cost of goods sold)).

Quality is persistent (high autocorrelation, leading to low-turnover strategies).

The *price of quality*, *i.e.*, the slope of the regression $\log(P/B) \sim \text{quality}$ (one could use the z -score instead of a logarithm; one can define the price of each quality component similarly) is lower during bubbles.

Value investing: the use of historical financial statement information to separate winners from losers
J.D. Piotroski (2002)

Value portfolios often contain many poorly performing stocks, their performance coming from a small number of strongly performing stocks. They can be improved with a quality measure such as the Piotroski score, a sum of binary variables measuring profitability (ROA, CFO, change in ROA, accruals), capital structure (change in leverage, change in liquidity (current ratio), new equity issuance), efficiency (change in gross margin, change in turnover). This works best for stocks with inefficient information dispersion (low volume/float, market capitalization, number of analysts, media coverage, availability of earnings forecasts, etc.).

Most of the performance is gained around earnings announcements.

Financial statement analysis also separate winners from losers in Brazil
A.B. Lopes and F.C. Galdi

Even in emerging countries such as Brazil, the Piotroski score can improve value portfolios.

Change in cash-holding policies and stock return predictability in the cross section
W.R. Sodjahn (2013)

The change in cash holdings (cash/assets) has predictive power on future returns, especially for low cash/assets firms, even after controlling for size, P/B, momentum, growth, liquidity (current ratio) and idiosyncratic volatility (standard deviation of the Fama-French residuals).

The accrual anomaly
P.M. Dechow et al. (2011)

Cash earnings are more persistent than accrual earnings. Many investors focus on earnings rather than cash: accruals, *i.e.*, the piece of earnings made up by accountants, do not seem to be correctly priced. High accruals are a good predictor of inventory write-downs, SEC enforcement actions, etc.

Earnings manipulations and expected returns
M.D. Beneish et al. (2013)

The M-score is a refinement of Sloan's accruals: it combines accruals with changes in receivables/sales, gross margin, proportion of other assets, sales, depreciation rate, SGA/sales, leverage. It predicts changes in accruals and the futures returns, even after controlling for size, momentum, value and accruals.

The detection of earnings manipulations
M.D. Beneish (1999)

Construction of the M-score, a probit model to predict manipulation (future SEC accounting enforcement action or media coverage) for non-financial US firms in the late 1980s.

Financial ratios and the probabilistic prediction of bankruptcy
J.A. Olson (1980)

The Ohlson model, a refinement of the earlier Altman model, is a logit model to predict the bankruptcy of US industrial firms in 1970–1976 using the following variables: size, liabilities/assets, current ratio, net income/assets, cash flow from operations/liabilities, change in net income. (There are 3 more variables, but they are not significant: working capital/assets, liabilities \geq assets (to allow for discontinuities), negative net income for the past 2 years.)

The quality dimension of value investing
R. Novy-Marx (2013)

Review of many quality measures – Graham score, Grantham quality (high ROE, low ROE volatility, low leverage), Greenblatt's return on invested capital (EBIT/tangible capital), Sloan's accruals, Piotroski's score, Novy-Marx gross profit/assets ratio – and how they can be combined with value factors (E/P, B/P,

EBIT/EV, where the enterprise value EV is market capitalization + liabilities – cash).

The tree of LIFO
R. Cahan et al. (DB, 2012)

Many accounting-derived investment signals have been suggested, capturing accounting manipulations (e.g., difference between cash and accrual earnings), overinvestment, return on capital, distress, etc. Classification trees (CART) can combine those factors, capturing non-linearities (default models do not distinguish between low-default-risk assets; many signals show an “inverted U” pattern, *i.e.*, extreme values are bad) and conditional payoffs (e.g., default models that do not include volatility work better for high-volatility stocks; different sectors (financials...) behave differently) and avoid overfitting through pruning.

Fitting them on a 5-year window captures changing market conditions and investor preferences (leverage is good/bad in risk-seeking/adverse environments).

The model can be used as a “prescreen” for non-quant investors, or as an overlay, a long-only or a long-short strategy.

Using link mining for investment decisions: extending the Black-Litterman model
G. Creamer (2013)

Directors and companies form a bipartite network. Many social network statistics (degree, degree centrality, closeness centrality, betweenness centrality, (normalized) clustering coefficient), etc. – measured on the bipartite graph, or the graph of directors, or the graph of companies) estimate in how many pies board members have their fingers, *i.e.*, they gauge corporate governance. [No recent data available.]

Analysts following companies (IBES data) can be used in a similar way.

Understanding hedge fund alpha using improved replication methodologies
J. Chen and M.L. Tindall (2013)

Hedge fund indices can be replicated using penalized regression (lasso) or state space models (linear models whose coefficients follow a random walk, with independent errors). Methods known to be suboptimal (stepwise regression, principal component regression, partial least squares (PLS)) are indeed suboptimal. (The authors do not list the factors retained by the model and their coefficients – they tend to be very noisy and rather unusable.)

The factors used also contain option-like assets:

- Buy-write index (buy the S&P 500 and write a (covered) near-term call);
- Synthetic call (buy a 3-month ATM call and sell it one month later; do not use market prices (?) but the Black-Scholes formula, with the VIX as implied

volatility, the 3-month T-bill as risk-free rate, and the S&P 500 dividend yield);

- Synthetic put.

Regularized hedge fund clones
D. Giamouridis and S. Paterlini

Two methods have been proposed to replicate hedge funds:

- Moment replication: try to match the second, third and fourth moments of the distribution of hedge fund returns and hope that the first moment (the returns) will follow;
- Factor replication.

Adding an L^1 penalty to factor replication

$$\begin{array}{ll} \text{Find} & \beta \\ \text{to minimize} & \|r - F\beta\|_2^2 + \lambda \|\beta\|_1 \\ \text{such that} & \mathbf{1}'\beta = 1 \\ & -1 \leq \beta \leq 1 \end{array}$$

creates smaller portfolios, with a lower turnover (an L^2 penalty would also reduce the turnover), and therefore a lower implementation cost.

Hedge fund replication: putting the pieces together
V. Weber and F. Peres (2013)

Hedge fund style classification labels have little value (the entropy of the conditional probabilities $P[\text{cluster}|\text{style}]$, where the clusters come from the k -means algorithm, is high).

Hedge funds can be replicated (even if we account for transaction costs and liquidity constraints) from a set of *dynamic risk factors* (rule-based trading strategies) and various regression techniques (ordinary least squares (OLS), weighted least squares (WLS), lasso, BIC model selection):

- Dollar-neutral equity portfolios (e.g., Topix vs S&P 500);
- Carry strategies (within each asset class, go long markets in backwardation, go short markets in contango);
- Equity risk parity;
- Long-short momentum strategies (for each asset class);
- Long currency futures;
- High yield (there are no corporate bonds, but one can use a portfolio of government bonds and equities, matching the duration and the equity beta).

Hedge fund returns should be *corrected for autocorrelation*.

Alpha factors in risk models
(M. Brown, CQAsia 2013)

Try to include your alpha, or its components, in the risk model – but if you already had similar risk factors, remove them. (Some people also suggest to rotate

the risk model to make it orthogonal to your alpha, so that the optimization reduces the risk orthogonal to your alpha, without reducing your alpha.)

Flight to safety
(T. Marsh, CQAsia 2013)

The “flight to safety” is problematic: since markets have to clear, if investors want to sell risky positions, someone else has to be willing to buy them – or, rather, the price has to drop enough for someone to be willing to buy them.

The market can be modelled with 10 investors, with varying risk aversions, 5 assets, with known expected returns, volatilities and correlations, in quantities and prices such that everyone holds an optimal portfolio and markets clear. After a crisis, the prices have changed, and the investors no longer hold an optimal portfolio; in addition, their risk tolerance has dropped, and their estimates of expected returns, volatilities and correlations have changed. Can they rebalance their portfolios? How should prices change to allow markets to clear?

Wealth transfer
(H. You, CQAsia 2013)

New equity issues (resp. share repurchase) are bad (resp. good) for existing investors: they should be included when computing the value of the firm.

Generalized risk-based investing
(T. Michel, CQAsia 2013)

Equal-weighted, minimum risk, risk parity, maximum diversification (all examples of “smart betas”, *i.e.*, alternatively-weighted indices) and equal-weighted portfolios are special cases of

$$\forall i, j \quad \frac{w_i^\alpha}{\sigma_i \beta} \text{MCR}_i = \frac{w_j^\alpha}{\sigma_j \beta} \text{MCR}_j$$

for various values of α and β .

An innovative look at corporate credit risk
G. Bonne (CQAsia 2013)

There are several models to estimate the probability of default:

- Structural (Merton) models, based on leverage, volatility of assets, drift of assets;
- Ratio analysis (works better with analyst forecasts instead of reported values);
- Text mining (using filings, transcriptions of meetings with investors, etc.) – the general sentiment lexicons do not work well in this context
- Combined models.

Generating Alpha from event-based investing
(D. Pope, CQAsia 2013)

Event studies should not limit themselves to computing the average return around an event, but should use a portfolio approach, trying to actually design and implement a strategy around those events.

- The announcement of activist investing raise the price two or three weeks before and after the announcement (13D filing, when they own more than 5% of the company).
- After a CEO change, accompanied with an improvement in quality (say, return on assets (ROA) ratio) in the next six months, the price rises for one year; the effect is less pronounced with CFOs (they are more often poached).
- There is a positive (resp. negative) reaction to dividend initiations and increases (resp. decreases).

The momentum of complicated firms (conglomerates) lags that of their peers.

Quant in Asia
(B. Lau, CQAsia 2013)

One can measure overcrowding as follows: take 120 factors, combine them in random ways, look at the performance of the corresponding strategies, and compare the portfolios in the top 10% – there is some overlap, but not that much.

Gearing into reverse
C. Natividade (DB, 2013)

The variance ratio

$$\text{VR}(q) = \frac{\text{Variance}(\text{returns on intervals of size } q)}{q \times \text{Variance}(\text{returns on intervals of size } 1)}$$

(for a random walk, it is a constant function of q) suggests there is a short-term (30 minutes to 2 hours) reversal. The reversal is more pronounced for assets with a low correlation to the VIX.

Colours of trend
C. Natividade et al. (DB, 2013)

How to build a trend-following strategy:

- Use trend (time series momentum) rather than (cross-sectional) momentum if your universe has few assets;
- Add liquidity constraints (e.g., 1% of the average daily volume);
- Compute each signal for 9 different window sizes, and aggregate them (unweighted average, first principal component, or using a sentiment indicator – more weight to recent value in adverse market conditions);
- Combine all the strategies, e.g., with equal weights, mean-variance optimization, equal-risk contribution, maximum diversification, residual momentum (regress the strategy returns against the first two principal components of the asset returns, rank the (cumulated?) residuals);

- For the variance matrix needed by the mean-variance optimization, use a regime-specific variance matrix (from the sentiment indicator);
- Fine-tune the rebalancing frequency;
- There is no evidence that stop-loss rules improve the strategy.

The signals used are the following direction signals (in $\{\pm 1\}$):

- **Velocity:** $\text{EWMA}(X, a) - \text{EWMA}(X, b)$ (you can use a Hodrick–Prescott filter instead of the exponential-weighted moving average, or any low-pass filter);
- Aggregate correlation;
- Auto-regression;
- Naive direction: $X_t - X_{t-a}$;

and strength signals (in $[0, 1]$ – they can be used as trade size or leverage):

- Hurst exponent;
- Mann-Kendall test;
- Auto-turbulence;
- Technical analysis recipes (RSI, VHF, etc.).

Pairing singles

C. Natividade (DB, 2013)

Pairs trading can be seen as a trend-following strategy for several assets:

- The two assets have started to diverge and we expect them to continue to diverge;
- The two assets are usually close, but they have started to diverge; we expect them to converge soon (cointegration was not mentioned).

Clock-watchers

C. Natividade (DB, 2013)

With the volume clock instead of the wall clock (divide the day into 96 blocks of equal volume, corresponding to 15 minutes of average volume), trend-following strategies (e.g., Mann-Kendall test; short- versus long-term moving average (5 vs 25 blocks); trend-following with structural-break-test-induced stop-loss) are less volatile.

Machine (un)learning

C. Natividade (DB, 2013)

Try the following machine learning algorithms:

- Ordinary least squares;
- Support vector machines (SVM);
- Penalized regression (not mentioned);
- Gaussian process regression (in R: `gptk`, `kernlab`, etc.);
- Recurrent neural nets (RSNNS in R, `neurolab` or `pybrain` in Python);
- Deep neural nets (not mentioned, and difficult to implement);
- Regime switching models (hidden Markov models, HMM);

to predict future returns from the following features:

- Velocity;
- Acceleration;
- **Mann-Kendall test:**

$$\sum_{j-i \in \{3, 5, 10, 20, 30\}} \text{sign}(x_j - x_i)$$

- **Auto-turbulence:**

$$(y_i - \mu)' \Sigma (y_i - \mu)$$

$$\Sigma = \text{Var } y$$

$$\mu = E[y]$$

$$y_t = (x_t, x_{t-1}, \dots, x_{t-30})$$

(you can restrict y_t to the components with a significant correlation with x_t);

- Volatility (3-month, exponentially weighted).

Switching styles

K. Chen and C. Natividade (DB, 2012)

There are three main FX strategies:

- **Carry:** buy (sell) the three G10 currencies with the highest (lowest) money market rate; rebalance every 3 months;
- **Momentum:** buy (sell) currencies that appreciated (depreciated) the most over the past year; rebalance every month (this works well when the volatility is either low or high);
- **Valuation:** buy (sell) the three cheapest (dearest) currencies (for the purchasing power parity, PPP); rebalance every 3 months.

Here are a few ways of combining them:

- Style momentum: increase the weight of the best-performing one;
- **Moment optimization** (generalizations of the minimum variance portfolio – but you need robust estimators of higher comoments: L -moments, shrinkage, etc.): maximize

$$E[X] - \lambda_1 \text{Var } X + \lambda_2 \text{Skew } X - \lambda_3 \text{Kurt } X;$$

- Other general portfolio construction procedures: maximum diversification, equal risk, etc.
- Regime switching, using a market sentiment indicator.

The **sentiment indicator** is the first principal component of the following factors:

- VIX;
- MSCI Financials/MSCI World;
- Difference between 30Y and 2Y asset swap spreads;
- Average of USD, EUR, JPY 3M5Y swaption volatilities;
- TED spread (interbank rates vs treasury bills);
- Spread between BAA corporate bonds and 10-year treasuries;
- Non-financial CDS spreads;
- FX implied volatility (CVIX);

- FX volatility slope (1Y vs 1M);
- FX volatility skew;
- Emerging markets sovereign risk.

***Nineteen dubious ways to compute
the exponential of a matrix,
twenty-five years later***
C. Moler and C. Van Loan (2003)

There are many algorithms to compute the exponential of a matrix, with problems regarding reliability (the precision of the result should be known, a warning should be issued if it is insufficient), stability, accuracy, efficiency: series methods (Taylor, Padé), scaling and squaring ($e^A = (e^{A/m})^m$, where $e^{A/m}$ is computed with a series), rational approximation (good if the eigenvalues are negative), ODE solvers (general solvers are very inefficient, but the algorithms can be specialized to this special case), characteristic polynomial (if you know it, e^{tA} is a polynomial in A), eigenvalues, matrix decompositions ($A = SBS^{-1}$, with B diagonal, Jordan, triangular or block-diagonal – find a good compromise between “close to diagonal” and “well-conditioned”: the blocks correspond to nearly-confluent eigenvalues), Trotter’s formula ($e^{B+C} = \lim_{n \rightarrow \infty} (e^{B/m} e^{C/m})^m$, with $B = (A + A')/2$, $C = (A - A')/2$, and e^A , e^C are computed using their eigenvalues), Krilov spaces ($\text{Span}\{v, Av, A^2v, \dots, A^mv\}$, with m small).

In R, use the `expm` package.

The *pseudo-spectrum* can help explain the behaviour of the matrix exponential:

$$\begin{aligned}\Lambda_\varepsilon(A) &= \{z : \|(zI - A)^{-1}\| \geq \varepsilon^{-1}\} \\ &= \{z : z \in \Lambda(A + E), \|E\| \leq \varepsilon\} \\ &= \{z : \|(A - zI)v\| \geq \varepsilon, \|v\| = 1\} \\ &= \{z : \|(zI - A)^{-1}\| \geq \varepsilon^{-1}\}.\end{aligned}$$

There are nice pseudospectra plots on the Pseudospectra gateway website.

***Visualizing and inspecting
large datasets with tableplots***
M. Tennekes et al. (2013)

The *tableplot* is very similar to the *lensplot*, but the rows correspond to aggregated values, not individual observations (for qualitative variables, it is a stacked area plot, for quantitative variables, just the plot of the average in each bin – use 100 bins).

An alternative to `describe` for data exploration: `tabplot`, `tabplotGTK`; use `ff`, `LaF` for larger datasets.

***DAKS: an R package for data analysis
methods in knowledge space theory***
A. Ünlü and A. Sargin

Item response theory (IRT) (`1tm`, `mokken` in R) is used in psychometrics to model (students’) abilities and (tests’) difficulties. Knowledge space theory (KST) fulfills the same role, but is deterministic and based on the

notion of partial order (on the set of skills: $i \rightarrow j$ if skill i entails skill j). The basic local independence model (BLIM) is a probabilistic KST, accounting for a non-zero probability of careless error and lucky guess (for each item). The inductive item tree analysis (IITA) algorithm builds a sequence of partial orders (start with $i \sqsubseteq_0 j$ if $j \rightarrow i$ has no counter-examples; then $i \sqsubseteq_L j$ if $j \rightarrow i$ has at most L counter-examples and does not add any non-transitivity to \sqsubseteq_{L-1}) and pick that closest to the data, *i.e.*, minimizing $\sum_{i \neq j} (b_{ij} - b_{ij}^*)$, where b_{ij} is the number of counter-examples of $j \rightarrow i$ and b_{ij}^* their expected number.

***Mining the web for the “voice of the herd”
to track stock market bubbles***
A. Gerow and M.T. Keane

Vocabulary uniformity in financial articles (FT) (smaller set of verbs – only positive ones; smaller set of nouns – focus on a smaller set of market events) can help detect crises: use the (windowed geometric mean of the) exponent in the power law distribution of word frequency. Verbs work better. In the bull market prior to a crisis, the distribution of frequencies of positive verbs is different (as measured by the Kullback-Leibler divergence) from the long-term distribution.

***Modeling movements in oil,
gold forex and market indices using
search volume index and twitter sentiment***
T. Rao and S. Srivastava (2012)

Google search volume (use a list of words, suggested by the “related searches”, and reduce the dimension with principal component analysis (PCA)) and twitter sentiment (same keywords) can predict future weekly returns (GLD, WTI, DJIA, Nasdaq, EUR), with “95% accuracy” (sic) – even though the sentiment estimator itself is only 80% accurate.

Critical truths about power laws
M.P.H. Stumpf and M.A. Porter (2012)

All too often, the omnipresence of power laws:

- Lacks statistical support;
- Lacks a mechanistic explanation: it could just be due to the central limit theorem and the aggregation (sum) of heavy-tail random variables;
- Provides no new insight.

A guide to vehicle routing heuristics
J.F. Cordeau et al. (2002)

List of a few VRP algorithms: Clarke-Wright savings heuristic, sweep algorithm (rotate a ray centered at the depot), adaptive memory (a kind of genetic search), and various tabu algorithms (random tabu duration, penalized objective, removal of unlikely (*i.e.*, long) edges, etc.).

***A new flexible direct ROC regression model
Detection of cardiovascular risk factors
by anthropometric measures***
M.X. Rodríguez-Álvarez et al. (2010)

The dependence of the ROC curve on covariates can be modeled with a GLM or GAM model (ROC-GLM, ROC-GAM).

Data stream mining: a practical approach
A. Bifet et al. (2011)

MOA is a Java library for machine learning on streaming data, an online analogue of Weka. Contrary to Vowpal Wabbit, it only allows a small number of features (hundreds), focuses on algorithms with theoretical guarantees, (mainly decision trees and ensemble methods), and can deal with time-changing models (*concept drift*).

1. Hoefding trees (VFDT, very fast decision trees) only look at each observation once, and split a node if the information gain and/or the difference between the best and the next best variable to split on exceeds some threshold (computed from *Hoefding's inequality*); this requires storing sufficient statistics at each node. To save memory, some nodes can be deactivated. To save time, the information gains are only computed every 200 observations. Quantitative variables can be approximated by histogram (quantile) summaries or a Gaussian approximation.

For prediction, one can use the majority class of a leaf, or use the sufficient statistics stored in that leaf in a naive Bayes classifier, or use whichever method worked best for that leaf in the past.

To estimate performance, use measures robust to unbalanced classes, such as *Cohen's kappa*.

2. Online bagging estimates several models in parallel (a small number, to keep resource usage low), including each observation $\text{Pois}(1)$ times (offline bagging corresponds to a binomial distribution, whose limit is a Poisson distribution). To improve online bagging:

- Include each observation $\text{Pois}(\lambda)$ times, with $\lambda > 1$, to increase weight diversity;
- Output a random class (sampled according to the predicted probabilities), instead of the majority one.

Online boosting is similar, but with $\text{Pois}(\lambda_{i,m})$ instead, for observation i and model m , with $\lambda_{i,1} = 1$, and $\lambda_{i,m+1}$ computed from $\lambda_{i,m}$ depending on whether observation i is classified properly by model m . Online boosting does not perform as well as online bagging, especially in the presence of changes.

Instead of separate models, *option trees*, *i.e.*, trees with option (non-deterministic) nodes (both branches are taken) are more memory-efficient.

Ensembles of classifiers of different sizes (say, 2^n , for various values of n – when the size exceeds the threshold, either delete the root (the oldest node) and all its children except one – or delete everything and start

afresh) perform better: smaller trees adapt quickly, and larger trees perform better when there is no change (use weights proportional to the inverse of the square of the error rate, estimated with an exponential moving average).

To avoid overfitting, one can use depth-limited trees, or trees built on distinct subsets of the attributes (all k -element subsets) and combined with a (softmax) neural network, trained alongside the trees.

3. To account for concept drift, one can:

- Fit the model on a moving window;
- Review the previous decisions, at each node, and check if we would still make the same decision;
- Check if the error rate decreases (it should, as long as the data-generation process does not change).

The data in the moving window can be used, not only to detect change, but also to update or refit the model. The windows can be:

- A fixed window of size W (reference) and the last W observations;
- The last W observations and the W observations immediately before;
- All the possible partitions of the previous W observations into two windows.

This last idea can be improved (ADWIN):

- Only consider $\log W$ windows (the past $\lfloor c \rfloor$, $\lfloor c^2 \rfloor$, $\lfloor c^3 \rfloor$, etc. observations);
- Use an approximate data structure (exponential histogram) to store the data needed;
- Instead of an actual statistical test, use a threshold, derived from Hoefding's inequality.

This can be combined with a Kalman filter. For Hoefding trees, look for changes at each node, but adjust the threshold for multiple testing.

***Introduction to stream:
a framework for data stream mining***
J. Forrest et al.

stream is an R package for machine learning on data streams, focusing on clustering (cluster the data into a large number of micro-clusters, online, then cluster the micro-clusters into macro-clusters, offline). The interface allows for other mining tasks (classification, itemset mining), but there is no implementation yet. The clustering algorithms from MOA are available.

***A general framework for observation-driven
time-varying parameter models***
D. Creal et al. (2008)

GAS (generalized autoregressive score) models are a unified framework to describe GARCH-like models (observation-driven models, *i.e.*, the time-dependence of the parameters comes from lagged, observed values –

as opposed to, say, stochastic volatility)

$$y_t \sim p(y|y_1, \dots, y_{t-1}; f_1, \dots, f_{t-1}; x_1, \dots, x_t; \theta)$$

$$f_t = \omega + \sum_i A_i s_{t-i} + \sum_j B_j f_{t-j}$$

$$s_t = S_{t-1} \frac{\partial \log p}{\partial f_{t-1}}$$

$$S_{t-1} = I \text{ or } S_{t-1} = E_{t-1} \left[\frac{\partial \log p}{\partial f_t} \frac{\partial \log p'}{\partial f_t} \right]^{-1}.$$

(This can be seen as a kind of smoothed or shrunk Kalman filter.)

Structural and topological phase transitions on the German stock exchange
A. Sienkiewicz et al. (2013)

The minimum spanning tree (MST) computed from the correlation of stock returns changes with time, sometimes a scale-free network (the degree distribution is a power law), sometimes a star-like network (idem, with one outlier).

Coupling between time series: a network view
S. Mehraban et al. (2013)

Some of the properties of a time series can be read from its visibility graph: the vertices are the timestamps, and there is an edge between i and j if $\forall k \in]i, j[$,

$$y_k < y_i + \frac{k-i}{j-i}(y_j - y_i).$$

It can be generalized to pairs of time series (normalize them both first): edge between i and j if

$$\forall k \in]i, j[\quad y_k \leq y_i + \frac{k-i}{j-i}(x_j - x_i)$$

or

$$\forall k \in]i, j[\quad y_k \geq y_i + \frac{k-i}{j-i}(x_j - x_i).$$

Dynamics of episodic transient correlations in currency exchange rate returns and their predictability
M. Žukovič (2013)

One can look for predicability in financial time series by testing if the correlation $C_{xx}(s)$ or the bicorrelation $C_{xxx}(r, s)$ of the log-returns

$$C_{xx}(r) = E[x_t x_{t-r}]$$

$$C_{xxx}(r, s) = E[x_t x_{t-r} x_{t-s}],$$

estimated on a moving window, is zero.

Dancing links
D.E. Knuth (2000)

Dancing links, *i.e.*, the remark that it is easy to remove *and add back* an element in a doubly-linked list, can be used to efficiently keep track of the current state in depth-first (backtracking) search. For instance, in the *exact set cover* problem (given a boolean matrix, find a set of rows with exactly one 1 in each column), one can encode the matrix as two doubly-linked lists (rows and columns), and descending in the tree just removes a column and one or several rows. This can be used in many satisfaction problems:

- Packing pentominoes on a chessboard: one row for each possible position of each individual pentomino, one column for each chessboard square, 1 if it is occupied;
- The n -queen problem: one row for each possible queen position, one column for each row, column or diagonal, 1 if it is controlled;
- Sudoku: one row for each possible decision of the form “digit i in (k, l) ”, columns for constraints: “there is a number in (k, l) ”, “there is an i in row k ”, “there is an i in column l ”, “there is an i in square m ”.

Will central counterparties become the new rating agencies?
C. Kenyon and A. Green (2012)

Overreliance on central counterparties could be as bad as overreliance on rating agencies: through collateralized trades, they transform credit risk into liquidity risk, but sharp price changes (that exceed the buffer in the margin account) and invalid prices (from previous transactions (we want future transactions) or from a model) still pose problem – seek more diverse price sources.

Network analysis of correlation strength between the most developed countries
J. Miśkiewicz (2012)

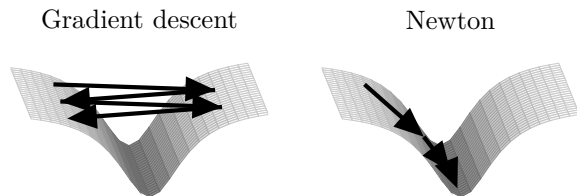
Instead of looking at the minimum spanning tree of a correlation matrix, *i.e.*, start with an empty graph and progressively add the most important edges, if they do not create cycles, until the graph is connected, do the opposite: start with a complete graph and delete the least important edges as long as the graph remains connected (the result is not a tree). [The first part of the article is a very confusing estimator of $\text{Cor}(\log x, \log y)$.]

Early prediction of movie box office success based on Wikipedia activity big data
M. Mestyan et al. (2013)

Editor activity of a Wikipedia entry is yet another (crowdsourced) data source, after Twitter and Google.

Deep learning via hessian-free approximation
J. Martens (2010)

Newton's method to minimize f approximates the objective as $q_\theta(p) = f(\theta) + \nabla f(\theta)'p + \frac{1}{2}p'Bp$ and minimizes it by solving $\nabla f(\theta) + Bp = 0$. Instead of exactly minimizing this second order Taylor expansion, minimize it approximately by a few conjugate gradient (CG) steps – the whole hessian B is not needed, only a few products Bp are (*truncated Newton*).



The article also lists a few modifications, useful for neural networks: damping (do not move too far: the quadratic approximation should remain good), back-propagation to compute products with the (Gauss-Newton approximation of the) hessian, mini-batches (large, and constant during the CG runs), termination condition for the CG, *no* backtracking (in the CG runs, keeping the parameter corresponding to the best value of f , as opposed to the last value, *i.e.*, the best value of q_θ), preconditioning (change of variables), start each CG iteration in the direction of the previous one.

**A practical guide
to training restricted Boltzmann machines**
G. Hinton (2010)

Advice on setting the many, many parameters in *contrastive divergence* (with a concise, but clear explanation).

**Deep neural networks for acoustic modeling
in speech recognition**
G. Hinton et al. (2012)

Review article on deep neural networks, Gaussian-Bernoulli restricted Boltzmann machines (GRBM), contrastive divergence.

High frequency trading and mini flash crashes
A. Golub et al. (2012)

Most mini flash crashes have a regulatory origin (intermarket sweep orders, ISO: the order is executed on the exchange with the best price first, which can deplete the order book (if the exchange is less liquid) and expose stub quotes).

Discrete optimization
P. Van Hentenryck (Coursera, 2013)

Lively overview of several approaches to optimization problems; as in the real world, for the programming assignments you are not told which algorithms will work for a given problem – your first ideas are likely to fail.

0. The first attempt to solve an optimization problem is often a *greedy algorithm*. For instance, for the traveling salesman problem, you can start at a node, take the nearest neighbour, and iterate; you could also (that is better) start with a 2-element cycle, add a vertex in the best position, and iterate. For the set cover problem (often represented as a binary matrix, one element per column, one set per row), you can take the sets in order, until all the elements are covered; better, you can take the largest sets first; even better, you can take the sets with the largest number of uncovered elements first – you can also simplify the problem, e.g., by removing dominated sets or checking if there are sets that have to be taken (because there is an element that is only in this set).

The knapsack problem can be solved exactly by *dynamic programming* (let $O_{j,k}$ be the value of the optimal solution with capacity k and items $\llbracket 1, j \rrbracket$) or (depth-first, best-first, etc.) *branch and bound* (to find a bound, relax the problem by removing the capacity constraint or, better, by replacing $x \in \{0, 1\}$ with $x \in [0, 1]$).

1. *Constraint programming* keeps track of the set of possible values (“domain”) of each variable, and tries to reduce those sets by “propagating the constraints”. For instance, in the 8-queen problem, “propagating the constraints” means “marking the cells that are no longer available”. It could be called “branch-and-prune”.

From an implementation point of view, a constraint is often an object with two methods, **satisfied?** and **propagate**; it only interacts with the domain store – constraints do not interact with one another.

There are dedicated **constraint propagation** algorithms for each type of constraint. For instance, to propagate an arithmetic constraint over integers $\sum a_i x_i = 0$, one can isolate one variable $x_i = \sum_{j \neq i} a_i^{-1} a_j x_j$ and find a lower and upper bound on x_i from the lower and upper bounds on the other x_j (a few iterations may be needed).

Most constraint programming languages or libraries allow reification (transforming a condition into a binary variable) and the use of decision variables as array index – these make constraint propagation a bit trickier.

Global constraints, e.g., **allDifferent**, could be expressed with elementary constraints, but if the solver is made aware of them, it can exploit their special structure (e.g., detect infeasibility earlier, or prune more) – sudoku can be solved efficiently in this way.

Scheduling problems often use more complicated global constraints.

The domains are usually intervals (bounds consistency), but one could consider arbitrary sets instead (domain consistency – more computationally expensive).

If the problem has a symmetry, symmetry-breaking constraints can reduce the search space a single coset.

For instance, a balanced incomplete block design (BIBD) is a $v \times b$ binary matrix with r ones in each row, k ones in each column, such that the scalar product of any 2 rows is ℓ . (This is used in experiment design and software testing: b new features, v tests, each with k features, each feature appearing r times, each pair of features is tested ℓ times.) The symmetries (row and column permutations) can be removed by requiring that the rows (and columns) be in lexicographic order.

Since constraints do not communicate with each other, *Redundant constraints* can speed up the computations: you could add the sum of the constraints, or (better), a linear combination of the constraints, with coefficients 1, α , α^2 , etc.; you could also find some less obvious property of the solution.

For instance, in the car sequencing problem (the cars to build have different option, but at most a out of b consecutive cars can have option o), the cars with a given option cannot be all at the end: there is a minimum number of them in each interval $[0, k]$ – these redundant constraints help the solver realize early that a partial solution is infeasible; it links the capacity and demand constraints.

If you have two very different models, you can give them both to the solver, and link them with constraints (*dual modeling*); for instance, for the 8-queen problem, there is one queen in each column, and we want to find the corresponding row numbers, but conversely, there is one queen in each row, and we want to find the corresponding column numbers.

The implementation of global constraints often relies on graph algorithms. For instance, for the `allDifferent` constraint, feasibility can be checked by the existence of a maximum matching in the bipartite graph of variables and values. (Feasibility is a special case of constraint propagation: it either leaves the domains unchanged, or sets them all to \emptyset .)

When exploring the tree of possible assignments, there are several strategies to choose which variable to instantiate and which value to set it to: often, the most-constrained variable (smallest domain and/or largest number of constraints) leads to earlier failures and more pruning. For instance, for the Euler knight problem, you can start in a corner, and then do the other corners. Instead of choosing the variable and then its value, you can do the opposite, *i.e.*, choose the value and then the variable (e.g., in the perfect square packing problem). Instead of choosing an actual value, you can just reduce the domain, e.g., by halving it – that is a weaker commitment (magic square, car sequencing).

2. For large problems, local search, which only provides a local optimum, is a scalable alternative to exact methods – but you need to pay close attention to the definition of the neighbourhoods, to how you explore them (randomly, exhaustively, heuristically), and implement them as efficiently as possible,

The simplest neighbourhood corresponds to changing

the value of a single variable, e.g., for a satisfaction problem (you are trying to minimize the number of breached constraints), choose a variable that appears in the largest number of violated constraints and a new value to minimize the number of violations.

The next simplest neighbourhood swaps the values of two variables – in particular, if there is an `allDifferent` constraint. For instance, in the TSP problem, the 2-opt neighbourhood corresponds to swapping two edges; it can be generalized to 3-opt (remove 3 edges and rewire the nodes to have another tour – contrary to 2-opt, there are many possible rewirings) or k -opt (Lin-Kerningham).

If the the number of breached constraints is too coarse a measure of how bad a solution is, you can use the “degree” of violation instead, e.g., in an arithmetic constraint, the difference between the lhs and the rhs.

When using local search to solve a constrained optimization problem, you can solve a sequence of satisfiability problems (e.g., for graph colouring, look for a solution with k colours, and progressively decrease k – you can use the solution with $k + 1$ colours, with one colour removed, as a starting point for the search for a solution with k colours), or build a sequence of feasible solutions (e.g., maximize $\sum_i |C_i|^2$, where C_i is the set of vertices with colour i , using Kemp chains to move from one feasible solution to another), or explore both feasible and infeasible solutions, by adding a penalty for breached constraints to the objective (e.g., change the colours of one node at a time and minimize $\sum 2b_i |C_i| - \sum |C_i|^2$, where b_i is the number of bad edges in colour i – it turns out that, in this example, local minima are feasible).

Heuristics, e.g., simulated annealing (with restarts, reheats, and perhaps even a tabu list) can help escape local minima.

Tabu search keeps a list of the last k states to avoid visiting them again; instead of the whole state, you can make some property of the state (especially if the state is complex), or the (inverse of the) move (e.g., swapping the values of variables i and j) tabu. There are many variants: aspiration (accept a tabu move if it is better than everything seen so far), intensification (store high-quality solutions and return to them periodically), diversification (when there has been no progress for some time, diversify the current state, e.g., by setting some of the variables to random values), strategic oscillation (change the proportion of time spent in the feasible and infeasible regions), *late-acceptance hill-climbing* (compare the candidate solution with the solution n steps earlier – only one comparison, and keeping the value of the solutions suffices) etc.

3. Mixed integer programs (MIP) can be solved by branch-and-bound, using a linear relaxation, and branching on the most fractional variables. Reformulating the problem can make the relaxation closer to the actual problem. For instance, the constraint $x \neq y$

the node you want to put in the fringe is already there is not sufficient, you also need to update its cost). The heuristic often comes from a relaxation of the problem.

2. Constraint satisfaction problems are similar (single agent, deterministic actions, fully observed state, discrete state space), but the state is not arbitrary: it is defined by variable assignments.

A binary constraint satisfaction problem (each constraint involves two variables) can be represented by a graph, with variables as vertices and constraints as edges (if the constraints are not binary, use a bipartite graph, with two types of nodes, variables and constraints). BFS does not perform well because it only considers actual solutions in the last layer; DFS fares slightly better, but it only realizes it made a mistake when the solution is complete. *Backtracking search* checks the constraints as early as possible to prune the search tree; it can be improved by changing the variable order (most constrained first), or the value order. It can be improved further by propagating the constraints via *arc consistency*: (for binary constraints) for each arc in the constraint graph, check if for each value x in the tail there is a valid value y in the head; reprocess the arcs if you remove something from their head. This can be generalized to k -consistency (each assignment of $k - 1$ variables can be extended to an assignment of all k) – 3-consistency (path consistency) is sometimes used.

The structure of the problem can help: independent sub-problems can be solved independently; if the constraint graph is a tree, process the variables in topological order to make them consistent, and assign them in the reverse order; if the graph is a tree after removing a small number of nodes, do an exhaustive search on those nodes and use the tree algorithm for the others (cutset conditioning); consider “mega-variables” (subproblems), overlapping to ensure consistency, and forming a tree (they should satisfy the running intersection property).

Iterative improvement algorithms can also be used: local search (e.g., the *min-conflict heuristic* changes a variable to the value that violates the fewer constraints – it works very well for the n -queen problem, $n = 10^7$); simulated annealing; genetic algorithms.

3. Adversarial search, *i.e.*, two-player, perfect-information, zero-sum, deterministic games (zero-sum means that there is only one utility function, one player tries to maximize it, the other tries to minimize it) uses the same ideas: the value of a state is the best achievable outcome from that state, against an optimal adversary (*minimax*); it is sometimes possible to prune the search tree (alpha-beta pruning). It is needlessly pessimistic against a non-optimal adversary (e.g., the PacMan ghosts move randomly). If there are resource limits, go down the tree as far as possible in the allotted time, and use an evaluation function, usually some linear combination of features, (if it is sufficiently deep, it should work well). Since the agent replans at each step, infinite loops (thrashing) are possible (decide on

plan A and go left, then decide on plan B and go right, etc.).

4. Expectimax replaces the worst-case in Minimax with the average case, *i.e.*, the uncertainty does not come from a perfect adversary, but from chance (equivalently, the adversary acts randomly). The values in the leaves are utilities, and the players maximize their expected utility.

5. In a Markov decision process (MDP), the effect of the agent’s actions are not deterministic. Formally, an MDP is the datum of a set S of states, a set A of actions, transition probabilities $T_{s,a,s'}$, rewards $R_{s,a,s'}$ and a start state. The aim is to maximize the expected (discounted) reward.

By introducing chance nodes for (s, a) pairs (q -states), one can use expectimax:

$$V(s) = \text{Max}_a Q(s, a) \quad \text{Max node}$$

$$Q(s, a) = \sum_{s'} T_{s,a,s'} (R_{s,a,s'} + \gamma V_{s'}) \quad \text{Chance node.}$$

The corresponding tree is huge, has a lot of redundancies, but we only want an optimal policy $S \rightarrow A$.

The value $V_k(s)$ of a state if the game ends after k steps can be computed iteratively (*value iteration*); the values V_k converge to V , but the policy usually converges faster.

Policy iteration starts with an arbitrary policy, computes the value of the states under that policy, finds the optimal policy for those values (expectimax, 1 step), and iterates until convergence.

6. Reinforcement learning tries to solve an MDP without knowing it:

- Passive reinforcement learning starts with a policy π and learns the corresponding state values (policy evaluation);
- Temporal difference learning directly learns the state values (at each step, update the estimated value of the state you are leaving), but not allow the the corresponding optimal policy (the rewards and the transition probabilities are needed);
- *Q-learning* does the same thing, but with the Q -values; it converges to the optimal policy, even if you do not follow it (off-policy learning).

To explore the state space, one could follow the current policy with probability $1 - \epsilon$ and act randomly otherwise (ϵ -greedy) or, better, explore areas whose badness has not been established yet (boost the Q -values by adding $k/\text{number of visits}$).

If there are too many states, approximate them by a set of features, and estimate the Q -values with linear combinations of features.

Semantic hashing

R. Salakhutdinov and G. Hinton (2007)

Deep neural networks (stacked restricted Boltzman machines (RBM)) whose deepest layer has a small

number of binary variables can model word count vectors more accurately than latent semantic analysis (LSA, *i.e.*, SVD-based low-dimensional approximations of the word-document co-occurrence matrix); the values of those binary vectors can be used as a semantic hash: texts that differ by only a few bits are similar.

***Do arbitrage-free prices
come from utility maximization?***
P. Siorpaes (2012)

In an incomplete market with no arbitrage, no-arbitrage prices (of non-traded assets) form intervals. For a given utility-maximizing investor, the set of threshold prices (sell if above, buy if below) is a sub-interval. As the investor's endowment varies, these intervals cover the no-arbitrage one.

***Forecasting intraday volatility
in the US equity market.
Multiplicative component GARCH***
R.F. Engle and M.E. Sokalska (2011)

Intraday volatility can be modeled as a product of three factors, $r_{it} = \sqrt{h_t} s_i q_{ti} \varepsilon_{ti}$:

- Daily volatility (from a factor risk model) h_t ;
- Deterministic diurnal volatility pattern $s_i = E[r_{it}^2/h_t]$
- GARCH residual volatility q_{ti} .

The results are more stable if you pool the stocks by industry.

***The European debt crisis:
defaults and market equilibrium***
M. Lagi and Y. Bar-Yam (2012)

Long-term sovereign interest rate and debt/GDP ratio are linked, and their evolution can help defaults (one can compute a market-implied critical debt/GDP ratio).

***Modeling dependence with C- and D-vine
copulas: the R-package CDVine***
E.C. Brechmann and U. Schepsmeier (2013)

Vine copulas describe the dependence between n variables with $n(n-1)/2$ bivariate copulas, using Bayes's formula

$$f(x_1, \dots, x_n) = f(x_1)f(x_2|x_1) \cdots f(x_n|x_1, \dots, x_{n-1}).$$

The K-function (Kendall function) is

$$k(t) = P[C(U, V) \leq t].$$

The *K-plot* plots the sample versus the theoretical K-function. The λ -function is

$$\lambda(t) = t - P[C(U, V) \leq t];$$

one can plot it (λ -plot, with the sample and theoretical λ -functions) or use it to reduce the copula fitting

problem to a curve fitting problem (for parameter estimation or goodness-of-fit tests).

$$F_i = \frac{1}{n-1} \sum_{j \neq i} \mathbf{1}_{x_j < x_i}$$

$$G_i = \frac{1}{n-1} \sum_{j \neq i} \mathbf{1}_{y_j < y_i}$$

$$H_i = \frac{1}{n-1} \sum_{j \neq i} \mathbf{1}_{x_j < x_i, y_j < y_i}$$

$$\chi_i = \frac{H_i - F_i G_i}{\sqrt{F_i(1-F_i)G_i(1-G_i)}}$$

$$\lambda_i = 4 \operatorname{sign}[(F_i - \frac{1}{2})(G_i - \frac{1}{2})]$$

$$\times \operatorname{Max} \{(F_i - \frac{1}{2})^2, (G_i - \frac{1}{2})^2\}$$

One can also look at the χ -plot, $\chi_i \sim \lambda_i$, where χ_i can be interpreted as a local measure of dependence ("local correlation") and λ_i as the signed distance to the median.

The CDVine package also provides tests and estimators.

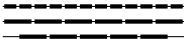
The fine-structure of volatility feedback
R. Chicheportiche and J.P. Bouchaud (2012)

The quadratic generalization of the GARCH model (QARCH)

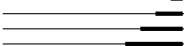
$$\sigma_t^2 = s^2 + \sum_{\tau > 0} L_\tau r_{t-\tau} + \sum_{\tau, \tau' > 0} K_{\tau, \tau'} r_{t-\tau} r_{t-\tau'}$$

accounts for a dependence of the volatility on the sign of the past returns (linear term) and effects at different time scales (off-diagonal terms). Special cases include:

- The effect of both 1- and 2-day returns;

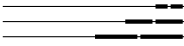
 squares of the returns
on those intervals

- The effect of the previous 1-, 5-, 20-day returns;


 squared returns

- Trends

 squared returns

 products

or

 products

***Beyond implied volatility:
extracting information from option prices***
R. Cont (1997)

In an incomplete market, the *state price density* (SPD) is not unique; to choose one, take that that minimizes hedging risk (for a given option) among all probability densities that give arbitrage-free prices (or among all densities, if hedging risk is more important than absence of arbitrage), or that closest to the historical density.

To estimate the SPD, use the option prices (or the volatility smile):

- Taylor expansion of the logarithm of the Fourier transform (cumulants, Edgeworth expansion);
- Hermite expansion of both the SPD and the payoff;
- Kernel estimator of the call option prices C to compute the density

$$e^{r(T-t)} \frac{\partial^2 C}{\partial K^2}$$

- Maximum entropy distribution under the constraints that the market prices are fair (without further constraints, it is often multimodal);
- Implied binomial trees (unstable);
- Mixture of log-normals (thin tails).

As an application, one can assume that the volatility is a function of the price, $\sigma = \sigma(S)$, estimate this function, compute the corresponding SPD, compare with another estimator (say, a kernel estimator), and use the difference as an investment signal.

The article also stresses the difference between expectation pricing (what accountants do) and no-arbitrage pricing.

Measuring capital market efficiency: global and local correlations structure
L. Kristoufek and M. Vosvrda (2012)

A low fractal dimension indicates herding; fractal dimension D and Hurst exponent H can be combined to form a “market efficiency measure”.

$$\begin{aligned} r_{xx}(0) - r_{xx}(\tau) &\underset{\tau \rightarrow 0}{\propto} \tau^{4-2D} \\ r_{xx}(\tau) &\underset{\tau \rightarrow \infty}{\propto} \tau^{2H-2} \\ r_{xx}(\tau) &= \text{Mean}_t[x_t x_{t+\tau}] \end{aligned}$$

Calibration of optimal execution of financial transactions in the presence of transient market impact
E. Busseti and F. Lillo (2012)

Finding the trading strategy (how much to trade in each 5-minute period, using the information available at that time, to trade a total quantity X during the day) to “minimize” the execution cost (the difference between the profit if everything had been traded at the open price and the actual profit – it is a random variable, so by looking at its expectation and variance, one can build a whole efficient frontier of trading strategies) can be formulated and solved as a quadratic problem.

Introduction to nloptr: an R interface to NLOpt
J. Ypma (2013)

NLOpt is a set of optimization algorithms, for constrained or unconstrained, global or local, gradient-free or not:

- Dividing rectangles (DIRECT, StoGO): not unlike Lipschitz optimization (if f is Lipschitz with constant k , then $\forall x \in [a, b] \ f(x) \geq f(a) - k(x - a)$,

- $f(x) \geq f(b) - k(b - x)$; by refining the intervals one can build a piecewise lower approximation of f), but without the knowledge of the Lipschitz constant;
- Controlled random search (CRS) with local mutation, improved stochastic ranking evolution strategy (ISRES): population-based random Nelder-Mead;
- Multi-level single linkage (local search, with many random or low-discrepancy restarts, that tries to avoid points leading to already-visited extrema);
- Constrained optimization by linear approximations;
- Bounded optimization by quadratic approximations;
- Principal axis (Brent);
- Nelder-Mead, Subplex;
- Method of moving asymptotes (MMA);
- Sequential quadratic programming (SLSQP);
- L-BGFS;
- Augmented Lagrangian (to turn a constrained optimization problem into a sequence of unconstrained ones).

An effective implementation of the Lin-Kernighan traveling salesman heuristic
L. Helsgaun

The 2-opt move for the traveling salesman problem (remove 2 edges and rewire in the only possible way) can be extended to k -opt moves (remove k edges and rewire in one of the many possible ways). The Lin-Kernighan heuristic uses the same idea, but extends a $(k - 1)$ -opt move into a k -opt move, but limits the choice of the k -th edge to the edge after the $(k - 1)$ st edge.

The traveling salesman problem: a case study in local optimization
D.S. Johnson and L.A. McGeoch (1995)

Among other algorithms, here are two heuristics to address the problem. The Clarke-Wright savings heuristics starts with a star path (return to the starting point after visiting each city) and progressively joins “nearby” cities. The Christofides heuristics constructs a minimum spanning tree, finds a minimum-length matching of its vertices of odd-degree (producing a graph all of whose vertices have even degree), and finds a Euler tour.

The late-acceptance hill-climbing heuristic
E. Burke and Y. Bykov (2012)

A variant of hill-climbing, as fast as, easier to implement than, and more robust than simulated annealing: keep a queue of past values; accept a candidate if it is better than the current solution or the solution in the queue L steps ago; put the candidate in the queue if it is better than the current solution.

The late-acceptance hill-climbing algorithm for the magic square problem
Y. Bykov

Another application of LAHC, to *magic frames* (to build a large magic square, constrain the problem some

more, and look for nested magic squares – the magic frame problems are smaller and independent).

(But there are constructive magic square algorithms: Siamese method, LUX method.)

The vehicle routing problem **T.G. Hinton (2010)**

Heuristics to solve the traveling salesman problem (TSP) include:

- The minimum spanning tree (MST) (pre-traversal order, after choosing a root);
- The *Christofides heuristic*: build the MST, find a minimum matching of the odd-degree vertices, find a Eulerian tour (*i.e.*, it visits all edges) of the resulting graph (all the vertices now have even degree), prune it to have a tour;
- Local search with k -opt neighbourhoods (remove k edges and rewire the graph into a tour in all possible ways); to speed up the search, use a neighbourhood list (only consider the $k = 40$ closest nodes of each node); with simulated annealing, select the starting temperature so that the average probability of accepting a detrimental move be 0.75.

Here are a few heuristics to solve the vehicle routing problem (VRP):

- Local search with λ -interchange neighbourhoods (swap λ vertices between two routes);
- n -opt- k : pick k routes, remove n edges, rewire in all possible ways;
- Iterated tour partition: solve the TSP problem, pick a node, start there, follow the tour until full capacity, start another tour, iterate until the end;
- Clarke-Wright (CW) heuristic (agglomerative greedy algorithm): start with 1-city routes, merge two routes, iterate – the routes to merge are those with the CW savings:

$$s(i, j) = c_{0,i} + c_{0,j} - c_{i,j}$$

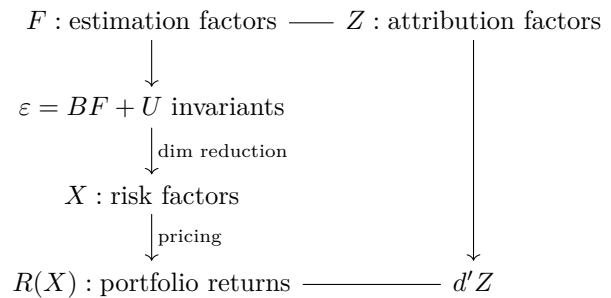
or their modified version

$$s(i, j) = c_{0,1} + c_{0,j} - \lambda c_{i,j} + \mu |c_{0,i} - c_{0,j}| + \nu \frac{d_i + d_j}{\bar{d}}$$

where \bar{d} is the average demand and $\lambda, \mu, \nu \in [0, 2]$ are chosen by exhaustive search (they try to avoid “ring routes” and favour large demands). It can be randomized: sort the possible moves from best to worst and take the i th one with probability $\frac{1}{q} \mathbf{1}_{i \leq q}$ or $p(1-p)^i$.

Factors on demand **A. Meucci (2010)**

Factor risk models are often used to both estimate and decompose risk. One can use different factors for estimation and attribution: given the joint distribution of (F, Z) or (X, Z) (often, simulated data), find d to minimize the discrepancy (variance, R^2 , VaR, etc.) between $R(X)$ and $d'Z$.



For instance:

- Statistical estimation factors (random matrix theory, RMT) and industry attribution factors.
- $X = Z$ = stock returns and $R(X)$ = price of an option on this stock defines “non-linear delta”

$$\underset{d}{\text{Argmin}} \text{CVaR}(\text{option returns} - d \times \text{stock return})$$

- Use detailed, regional (numerous) estimation factors and (fewer) global attribution factors;
- Style analysis (maximize the R^2 between the portfolio returns and a linear combination of reference strategies)
- Risk attribution to the alpha components: use portfolios associated to the components of the alpha as attribution factors and minimize the variance of the residuals.

Risk management **for central bank foreign reserves** **ECB (2004)**

3. Foreign reserves management is similar to assets and liabilities management (ALM), evaluated on a tree modeling the evolution of short rate and exchange rates, with a few differences:

- The portfolio is often separated into liquid and non-liquidity-constrained parts;
- Choosing a base currency is problematic;
- Constraints of the form

$$\text{reserves} \geq 12\text{-month debt repayments} + \text{imports}$$

are imposed on the liquid portfolio;

- If the distribution of final wealth is not satisfactory, it can be “shaped” by adding (several) value at risk (VaR) constraints.

4. Foreign reserves are often excessively constrained. If you can tolerate short-term underperformance, add corporate bonds; relaxing the (liquidity, currency, duration, credit) constraints, and allowing for a larger universe, moves the efficient frontier: you can gain 1.5% in expected returns or risk – assuming that your estimates of the various market parameters are correct. However, as you diversify your portfolio, measuring risk becomes trickier.

7. The conflicting targets, wealth preservation (in the home currency) and liquidity preservation (ability

to pay in foreign currencies), call for different benchmarks. One can use robust (worst-case) optimization

$$\underset{w' \geq 0}{\text{Max}} \underset{s, b}{\text{Min}} [w' \mu - \lambda_{s, b} (w - w_b)' V_s (w - w_b)]$$

to address multiple benchmarks b , multiple scenarios s (normal and hectic, with the same correlation, but different volatilities, estimated with robust (shrinkage) estimators), and different “weights” (risk aversions) $\lambda_{s, b}$ in each case. The problem can be reformulated as a linear problem with quadratic constraints. For non-Gaussian returns, directly optimize the expected utility with a set of scenarios.

10. To account for counterparty risk (credit risk), do not simply look at the credit ratings, but check:

- The probability of default $P[\text{default}]$;
- The expected and unexpected loss (given default),

$$\text{EL} = E[\text{loss}|\text{default}]$$

$$\text{UL} = \sigma[\text{loss}|\text{default}];$$

- The portfolio expected loss $\sum_i \text{EL}_i$, the portfolio unexpected loss

$$\text{UL}' \rho \text{UL}$$

where ρ is either the default correlation (“default” is a Bernoulli random variable) or the correlation of the asset returns.

As usual, the expected and unexpected portfolio loss can be decomposed into a sum of contributions and/or used in portfolio optimization.

13. Instead of decomposing the risk (ex ante), one can decompose the *change* in risk (ex post), either as

$$\begin{aligned} \Delta \text{VaR} = & \text{VaR}(w_t, p_{t-1}, t-1) - \text{VaR}(w_{t-1}, p_{t-1}, t-1) \\ & + \text{VaR}(w_{t-1}, p_t, t-1) - \text{VaR}(w_{t-1}, p_{t-1}, t-1) \\ & + \text{VaR}(w_{t-1}, p_{t-1}, t) - \text{VaR}(w_{t-1}, p_{t-1}, t-1) \\ & + \text{rest} \end{aligned}$$

or

$$\begin{aligned} \Delta \text{VaR} = & \text{VaR}(w_t, p_t, t) - \text{VaR}(w_{t-1}, p_t, t-1) \\ & + \text{VaR}(w_{t-1}, p_t, t-1) - \text{VaR}(w_{t-1}, p_{t-1}, t-1), \end{aligned}$$

where w are the portfolio weights, p the market data (prices) and t the time. This helps understand why the risk limits are breached – to avoid blindly selling positions.

17. To decompose the risk of a portfolio, use risk factors directly interpretable by portfolio managers (even if they are not independent – for a more modern approach, check Meucci’s *factors on demand*). For instance, for a fixed income portfolio, use use shift, slope and curvature factors; for an equity portfolio, use size, value (P/E, P/B, dividend yield) and momentum (earnings revisions, price reversals, earnings torpedo (high expected earnings and price drop)) factors.

18. The performance of a fixed income portfolio can be decomposed into time, curve and spread (*i.e.*, benchmark) components:

$$\begin{aligned} \Delta &= P(t, t) - P(t-1, t-1) \\ &= B(t, t) - B(t, t-1) && \text{time} \\ &\quad + B(t, t-1) - B(t-1, t-1) && \text{curve} \\ &\quad + \text{rest} && \text{spread} \end{aligned}$$

where

$P(\text{curve}, \text{time})$ = portfolio value

$B(\text{curve}, \text{time})$ = benchmark value

19. The PnL of fixed income portfolios can be decomposed into benchmark, duration, allocation and selection components, where “duration” captures the effect of the difference between the duration of the benchmark and that of the portfolio. The factors used to explain the PnL should be portfolio-dependent.

20. Liquidity crises can be modeled as follows:

- Random inflows and outflows occur daily, and have to be addressed by the overnight tranche of the portfolio;
- If the overnight tranche exceeds some threshold ON^* , the excess is moved to the weekly tranche;
- If the weekly tranche exceeds some threshold W^* , the excess is moved to the optimized portfolio;
- Every week, the overnight tranche is replenished, up to the target ON^* , from the weekly tranche.

Using simulations (e.g., the cashflows follow an autoregressive (AR) process) one can estimate the probability of a liquidity crisis, as a function of the target levels ON^* and W^* .

21. Choose your benchmark wisely: if it only depends on one parameter, duration, and if high duration means high returns, choose the maximum duration so that

$$\text{VaR}_{95\%} \geq \frac{1}{2} \times \text{risk-free returns}$$

(or some other threshold – the reference rate could be a constant rate, or a short-term rate minus a spread).

***The expectation maximization algorithm:
a short tutorial***
S. Borman (2009)

Let x denote the observed data, z the hidden data and θ the parameters to estimate. We want to maximize

$$L(\theta) = \log p(x|\theta) = \log \sum_z p(x, z|\theta),$$

iteratively. The improvement brought by θ on the pre-

vious estimate θ_n is

$$\begin{aligned}\Delta(\theta|\theta_n) &= L(\theta) - L(\theta_n) \\ &= \log \sum_z p(x, z|\theta) - \log p(x|\theta_n) \\ &= \log \sum_z p(z|x, \theta_n) \frac{p(x, z|\theta)}{p(z|x, \theta_n)} - \log p(x|\theta_n) \\ &\stackrel{\text{Jensen}}{\geq} \sum_z p(z|x, \theta_n) \log \frac{p(x, z|\theta)}{p(z|x, \theta_n)} - \log p(x|\theta_n) \\ &= \sum_z p(z|x, \theta_n) \log \frac{p(x, z|\theta)}{p(z|x, \theta_n)p(x|\theta_n)}.\end{aligned}$$

Since $L(\theta) \geq L(\theta_n) + \Delta(\theta|\theta_n)$, with equality if $\theta = \theta_n$, a value of θ that improves the right handside will also improve the left. After dropping the terms constant with respect to θ , we end up maximizing

$$\sum_z p(z|x, \theta_n) \log p(x, z|\theta) = E_{Z|X, \theta_n}[\log p(x, z|\theta)].$$

The single formula

$$\theta_{n+1} = \underset{\theta}{\operatorname{Argmax}} \sum_z p(z|x, \theta_n) \log p(x, z|\theta)$$

is traditionally (and confusingly) decomposed into E and M steps:

- Compute the conditional probabilities $p(z|x, \theta_n)$, one for each value of z (E step);
- “Compute” the function $\sum_z p(z|x, \theta_n) \log p(x, z|\theta)$ – it is a function of θ (E step);
- Find the value θ_{n+1} of θ that maximizes it (M step).

The generalized expectation maximization (GEM) algorithm does not use $\theta_{n+1} = \underset{\theta}{\operatorname{Argmax}} \Delta(\theta|\theta_n)$ but merely $\theta_{n+1} \in \{\theta : \Delta(\theta|\theta_n) > 0\}$; this is sufficient to ensure convergence.

A quasi-Newton acceleration for high-dimensional optimization algorithms H. Zhou et al. (2009)

The EM (expectation-maximization) iteration can be written $x_{n+1} = F(x_n)$ and (hopefully) converges to a fixed point of F . One can replace it with Newton’s method to solve $G(x) = 0$, where $G(x) = x - F(x)$: $x_{n+1} = x_n - G'(x_n)^{-1}G(x_n)$, i.e.,

$$x_{n+1} = x_n - (I - F'(x_n))^{-1}G(x_n).$$

To facilitate the inversion, one can use a low-rank approximation of $F'(x_n)$. The *quasi-Newton method* uses another approximation of $F'(x)$: sufficiently close to the fixed point x_∞ ,

$$F^2 x_n - F x_n \approx F'(x_\infty)(F x_n - x_n).$$

The smallest matrix $\widehat{F'(x_\infty)}$ satisfying those equations for the last q observations is an estimate of $F'(x_\infty)$.

The speed of Shor’s algorithm J.V. Burke et al. (2007)

Shor’s r -algorithm is a generalization of steepest descent ($\gamma = 0$) and conjugate gradient ($\gamma = 1$) to find a minimizer of a (non-necessarily smooth) function f

$$\begin{aligned}x_{n+1} &= x_n - t_n B_n B_n' f'(x_n) \\ t_n &= \underset{t}{\operatorname{Argmin}} f(x_{n+1}) \\ r_{n+1} &= B_n'(f'(x_{n+1}) - f'(x_n)) \\ B_{n+1} &= B_n(I - \gamma r_{n+1} r_{n+1}')\end{aligned}$$

with good performance for non-smooth functions (in this case, f' is a subgradient).

The R Journal (2013)

This issue contains more articles than usual; among others: `RTextTool` provides a unified interface to supervised classification algorithms for text processing (`svm`, `glmnet`, `maxent`, `sldca`, `ipred::bagging`, `caTools::LogitBoost`, `randomForest`, `nnet`, `tree`); “generalized simulated annealing” (`genSA`) may be faster than `DEoptim` or `rgeoud`; `ftsa` analyzes functional time series (e.g., a yield curve, a volatility surface); qualitative comparative analysis (QCA) is another name for association rules analysis in the social sciences (small datasets), often with fuzzy sets; many packages to manipulate maps (`ggmap`, `osmar`, `OpenStreetMap`, `RGoogleMaps`); applications of sparse matrices (`Matrix`) to fit GEE models and spatial probit models; etc.

To promote/enable reproducibility and productionalization, the R package manager should account for versioning (version dependencies are currently declared, checked, but not used), like Gentoo or NPM – but this would require the ability to install and load, simultaneously, several versions of the same package.

The generalized simulated annealing algorithm in the LEED search problem E.R. Correia et al.

Generalized simulated annealing (GSA) is simulated annealing on \mathbf{R}^D , with the following visiting distribution

$$g(\Delta x) \propto \frac{1}{\left(1 + (q-1) \frac{(\Delta x)^2}{T^{2(3-q)}}\right)^{\frac{1}{q-1} + \frac{D-1}{2}}},$$

cooling scheme $T(t) \propto t^{1-q}$ and acceptance probability

$$p(\Delta E) = \left(1 + (q-1) \frac{\Delta E}{T}\right)^{-\frac{1}{q-1}}.$$

Fast simulated annealing ($q = 2$, Cauchy visiting distribution) performs well.

Here are some of the topics mentioned:

- Instead of finding the portfolio weights maximizing the expected utility, one can smooth the expected utility and look for the center of a region (in the portfolio weights space) where it is high: *probabilistic utility* formalizes this idea and computes the expected weights w for the distribution

$$p(w) \propto \exp E[\text{utility}(\text{wealth}(w))];$$

- Part of the volatility smile is due to numeric imprecisions (far from the strike, the price is almost independent of the implied volatility);
- On a toy model of asset returns (say, ARMA), one can explicitly compute the Sharpe ratio of simple strategies (say, a moving average);
- Markov chains can model the evolution (survival) of exchange-traded derivatives, by looking at their (\log_{10} -binned) volume;
- Performance attribution;
- Implied expected returns can be computed from a *set* of mean-variance optimal portfolios, e.g., capital-weighted (market), equal-weighted ($1/N$), equal risk contribution, maximum diversification ($w' \sqrt{\text{diag } V} / \sqrt{w' V w}$), and risk efficient (the tangency portfolio, under the assumption that the returns are proportional to the downside deviation);
- The high-frequency-based volatility (HEAVY) model

$$h_t = \text{Var } r_t = \omega_1 + \alpha_1 V_{t-1} + \beta_1 h_{t-1} + \lambda r_t^2$$

$$\mu_t = E[V_t] = \omega_2 + \alpha_2 V_{t-1} + \beta_2 \mu_{t-1}$$

r_t : (daily) returns

V_t : realized volatility, measured from high-frequency data;

- Change-point analysis on a time series of independent observations can be performed using cluster analysis (agglomerative or divisive) with time constraints (to ensure that the clusters are contiguous – this is not unlike key detection in music), by maximizing

$$\mathcal{E}(X, Y, \alpha) = 2E|X - Y|^\alpha - |X - X'|^\alpha - |Y - Y'|^\alpha;$$

this is implemented in the **ecp** package;

- Cluster risk portfolios assign the same risk contribution to each cluster (either user-defined clusters, e.g., asset classes, or those resulting from some clustering algorithm), and then to each asset inside each cluster;
- Many models predict the yield curve or the S&P 500 from economic variables; combining them, with weights inversely proportional to the past performance (mean squared error, with older values discounted more), improves performance – but penalized models, with all the variables, do even better;
- Network analysis of interbank overnight loans, from SEC filings, can identify those whose failure can lead to a crisis (high centrality);

- Communities in a venture capital (VC) network influence performance;
- Remortgaging can be formulated as a game between lender and borrower;
- Crashes can be predicted with the log-linear power law (LPPL) model;
- Behavioural portfolios (or goal-based optimal portfolios) are mean-variance optimal portfolios, with a constraint on the value-at-risk.

The estimation of covariance matrices was a recurring topic:

- Factor models can also be used to decompose the cross-sectional volatility;
- When estimating a variance matrix, rescale the volatility with the (smoothed) VIX to make it stationary;
- Use robust correlation estimators (MCD for market outliers, robust pairwise correlation for specific risk outliers): to avoid believing in diversification when it is not there; for dynamic alerts (plot the Mahalanobis distance versus time with a (say) $\pm 3\sigma$ band);
- Principal volatility component (PVC) analysis looks for no-ARCH portfolios (not unlike independent component analysis, which looks for non-Gaussian portfolios);

A few new packages were presented: **RND** to estimate the risk-neutral density from option prices, **modopt** (not currently available) to specify optimization (and even stochastic optimization) problems as strings (unfortunately not as expressions, as Matlab's CVX allows), **greek**s, **MTS** (unavailable) for sparse modeling (sparse parametrization or penalized estimation) of multivariate time series (VAR, VARMA, etc.).

From a programming point of view, besides **Rcpp** (to access QuantLib functions or objects not available in **RQuantLib**, to use linear algebra libraries such as **Armadillo**), **quantstrat** (to build and test investment strategies), or **data.table**, **rzmq** (an interface to ZeroMQ, an easy-to-use and language-agnostic networking library) and **scidb** were also mentioned.

Efficient indexation

An alternative to cap-weighted indices

N. Amenc et al. (2010)

Compute the *tangency portfolio* by assuming that the returns are proportional to the (rank of the) downside risk (since investors require higher returns to invest in riskier stocks) and using principal component analysis to estimate the variance matrix; shrink towards the $1/N$ portfolio by adding minimum and maximum weight constraints, $1/2N \leq w \leq 2/N$; do not rebalance every quarter but only when the absolute value of the changes in weights exceed 50%.

Scalable robust covariance

and correlation estimates for data mining

F.A. Alqallaf et al. (2002)

Robust estimators of the variance matrix are often

computationally intensive ($O(e^{dn^2})$):

- Minimum covariance determinant (MCD): take 90% of the data (at random), compute the covariance matrix, iterate, keep the matrix with the smallest determinant;
- Minimum volume ellipsoid (MVE): take 90% of the data, find the smallest ellipsoid containing those points, iterate, keep the ellipsoid with the smallest volume (MCD can be seen as a constrained MVE: the ellipsoids are those defined by the sample mean and variance);
- Fast MCD: instead of taking a different, random, independent sample at each iteration, take the 90% of the data with the smallest Mahalanobis distance.

Faster estimates can be obtained by using robust pairwise correlation estimators (but you need to fix the correlation matrix afterwards):

- Rank correlation;
- Find the robust scale s_j and location m_j of each variable j ,

$$\sum_i \psi\left(\frac{x_{ij} - m_j}{s_j}\right) = 0,$$

where ψ is the Huber function $\psi(x) = \text{Min}(\text{Max}(-c, x), x)$ or the sign function $\psi(x) = \text{sign}(x)$ and take the correlation $\text{Cor}(y_{.j}, y_{.k})$ of the transformed data $y_{ij} = \psi\left(\frac{x_{ij} - m_j}{s_j}\right)$ (Huber correlation, quadrant correlation).

Propagation of outliers in multivariate data F. Alqallaf et al.

The contamination models used to develop robust estimators rarely include *componentwise outliers* – the resulting estimators have a poor breakdown, in high dimension, in presence of such outliers.

Portfolio optimization with mental accounts S. Das et al. (2010)

It is easier for investors to specify, for each of their goals (retirement, bequest, etc.) the minimum success probability than the overall risk aversion. Each goal corresponds to a subportfolio, optimized with a value-at-risk (VaR) constraint. Under a Gaussian assumption, those portfolios are mean-variance efficient; if short sales are allowed, the resulting portfolio is still optimal (if they are not, you only lose a few basis points).

Options and structured products in behavioural portfolios S. Das and M. Statman (2013)

Options have a role to play in behavioural portfolios (VaR-constrained portfolios), even if they are overpriced (by up to 20%).

Threshold cointegration: overview and implementation in R M. Stigler (2012)

The cointegration relation can be written in VECM (vector error correction model) form,

$$\Delta Z_{t+1} = c + \lambda' Z_t a + B \Delta Z_t + \varepsilon,$$

where $\lambda' Z_t$ are the cointegration relations (e.g.,

$$\lambda = \begin{pmatrix} 1 \\ -\beta \end{pmatrix}, \quad Z_t = \begin{pmatrix} X_t \\ Y_t \end{pmatrix}$$

and a is the mean-reversion speed). It can be extended to a threshold model by having the mean-reversion speed a depend on the current regime (the regime could correspond to the position of $\lambda' Z_t$ wrt some thresholds). This is implemented in the `tsDyn` package.

The misleading value of measured correlation B.M. Damghani et al. (2012)

Correlation is rarely a good measure of how linked two assets are, and cointegration should often be preferred. For instance, its assumptions (linear, synchronous, no outliers) are seldom satisfied; it is not valid if the spread between two underlyers follows a mean-reverting process (e.g., WTI and Brent – if you have a notion of “spread”, correlation is a bad idea), or for companies sharing the same, fixed-size market, or for commodities with an equity mirror (e.g., oil and BP), or for assets with contagion/domino effects (credit derivatives), or for overlays.

Instead of replacing the correlation model

$$\begin{aligned} \frac{dx}{x} &= \sigma_1 dW_1 \\ \frac{dy}{y} &= \sigma_1 \rho dW_1 + \sigma_2 \sqrt{1 - \rho} dW_2 \quad dW_1 \perp dW_2 \end{aligned}$$

with the cointegration (Ornstein-Uhlenbeck) one

$$\begin{aligned} \frac{dx}{x} &= \sigma_1 dW_1 \\ \frac{dy}{y} &= \theta(\mu + x - y)dt + \sigma_2 dW_2 \quad dW_1 \perp dW_2 \end{aligned}$$

one could try a hybrid model

$$\begin{aligned} \frac{dx}{x} &= \sigma_1 dW_1 \\ \frac{dy}{y} &= \theta(\mu + x - y)dt + \sigma_2 dW_2 \\ \text{Cor}(dW_1, dW_2) &= \rho. \end{aligned}$$

Improving portfolio selection using option-implied volatility and skewness V. DeMiguel (2012)

With a continuum of option prices, one can compute the model-free implied volatility (MFIV)

$$\text{MFIV}_{[t,T]} = \sqrt{EQ[e^{-r(T-t)} R_{[t,T]}^2]}$$

(or the other implied moments) and, assuming that MFIV/realized volatility is constant, forecast the future realized volatility – it is a better forecast of future volatility than historical volatility. Use implied volatility and implied skewness (either the model-free implied skewness, defined as the MFIV, or the spread between the IV computed from puts and calls) in your alpha to increase performance (Sharpe ratio, etc.)

***High-quality topic extraction
from business news explains
abnormal financial market volatility***
R. Hisano (2012)

Analysis of a huge set of business news suggests that news do have an impact on trade *volume*: for the news associated to a given company, discard all stock-related news, use *latent Dirichlet analysis* (LDA) for form 100 topics (sets of words, each word is in exactly one topic), and use penalized regression to try to predict future returns.

***Factor-risk-constrained mean-variance
portfolio selection: formulation
and global optimization solution approach***
S. Zhu (2011)

Constraints on the contribution of risk factors to the portfolio risk (a quadratic term) make the portfolio optimization problem non-convex. For up to a few hundred assets, it is amenable to branch-and-bound, with a semi-definite relaxation.

Measuring financial market stress
K.L. Kliesen and D.C. Smith (2010)

One can build a “financial stress index” by looking at (weekly) interest rates (government bonds with various maturities, TIPS, corporate bonds with various credit ratings), spreads (Libor-OIS, TED), and indices (emerging market bonds, VIX, bond volatility, equities) and taking the first principal component (of the demeaned, normalized time series) (the coefficients (“loadings”) are only estimated once, when the index is first constructed; they do not change with time).

***Stability analysis of financial contagion
due to overlapping portfolios***
F. Caccioli et al. (2012)

When banks hold diversified portfolios, those portfolios overlap; a bankruptcy triggers a flash sale and a devaluation of the assets, thereby impacting other banks. Contagion is made worse by overcrowding (ratio of the number of banks by the number of assets).

***Cascading failures in bipartite graphs:
model for systemic risk propagation***
X. Huang et al. (2013)

Idem.

***A model of market limit orders
by stochastic PDEs, parameter estimation,
and investment optimization***
Z. Zheng and R.B. Sowers (2012)

The limit order book can be modeled with a (system of) *stochastic PDEs* for mid-price (a function of time) and volume (a function of price and time – positive volumes for supply, negative for demand, and the volume at the mid-price is zero).

***Keynes meets Markowitz: the tradeoff
between familiarity and diversification***
P. Boyle et al. (2009)

Knightian uncertainty (uncertainty about the parameters of probability distributions) can be introduced in portfolio construction as follows: asset returns are Gaussian, $X \sim N(\mu_0, V)$; their variance V is known but their mean μ_0 is not; we only have an estimate $\mu \sim N(\mu_0, A)$, whose precision (ambiguity, A) is known; the investor wants to maximize the expected returns, with penalties for risk and ambiguity. The article uses a robust optimization instead:

$$\text{Max}_w \text{Min}_{\mu_0} \left\{ w' \mu_0 - \lambda_1 w' V w, \frac{(\mu - \mu_0)^2}{\text{diag } A} \leq \lambda_2 \right\}.$$

***Improved performance by constraining
portfolio norms: a generalized approach to
portfolio optimization***
V. DeMiguel et al. (2007)

Adding an L^2 penalty to the portfolio optimization problem

$$\text{Min}_w \{ w' V w + \lambda \|w\|_2^2, w' \mathbf{1} = 1 \}$$

is equivalent to shrinking the covariance matrix V to $V + \lambda I$; one can equivalently use a hard constraint

$$\text{Min}_w \{ w' V w, w' \mathbf{1} = 1, \|w\|_2 \leq \nu \}.$$

Adding an L^1 constraint $\|w\|_1 \leq \ell$ restricts the leverage to ℓ ($\ell \geq 1$, e.g., 1.3 for 130/30 portfolios).

Applying the conjugate gradient algorithm to the $1/N$ portfolio to minimize the variance produces a sequence of $N - 1$ “partial minimum variance portfolios”.

***Stock return serial dependence and
out-of-sample portfolio performance***
V. DeMiguel et al. (2010)

Use a VAR (or NAR – non-parametric auto-regressive – *nearest neighbour regression*, using the 50 days closest to today’s performance) model to build a momentum portfolio (use the conditional variance and mean, $V, \mu | X_{t-1}$; it will also capture lead-lag relations; you may want to add an L^1 constraint (or penalty) to reduce turnover.

The adaptive lasso and its oracle properties
H. Zou (2006)

Prefer the adaptive lasso (the coefficients of the L^1 penalties are variable-specific, e.g., $\lambda/\hat{\beta}_{ols}^\gamma$, with $\gamma > 0$) to the lasso: it makes variable selection consistent. In R, it is implemented in `lqa::adaptive.lasso`.

A kernel statistical test of independence
A. Gretton et al.

Covariance is not sufficient to test for independence, but in a universal reproducing kernel Hilbert space (RKHS), it is. “Universal” means dense in $\mathcal{C}_b(X, \mathbf{R})$ – in practice, you will choose a kernel and hope that it is not too far from universal. The norm of the covariance operator is

$$\text{HSIC} = E[k(x_1, x_2)\ell(y_1, y_2)] + E[k(x_1, x_2)]E[\ell(y_1, y_2)] - 2E_{x_1, y_1}[E_{x_2}[k(x_1, x_2)]E_{y_2}[\ell(y_1, y_2)]]$$

and it can be estimated as

$$\frac{1}{m^2} \text{tr}(KHLH)$$

where

m = number of observations

$k_{ij} = k(x_i, x_j)$

$\ell_{ij} = \ell(x_i, x_j)$

$$H = I = \frac{1}{m} \mathbf{1}\mathbf{1}'$$

k, ℓ : kernels on X and Y .

One can compute the asymptotic distribution of this estimator under the null hypothesis $P_{xy} = P_x P_y$ and estimate the critical value for a statistical test (or to convert the test statistic to a p -value).

New approaches in visualization of categorical data: R package extracat
A. Pilhöfer and A. Unwin (2013)

Use barcharts overlaid with mosaic plots (`rmb`) and interactive (iWidgets, iPlots) parallel graphs (`cpcp`), with the variables and values reordered to reduce overplotting.

Visualizing association rules: introduction to the R-extension package arulesViz
M. Hahsler and S. Chelluboina

Here are a few ways to visualize association rules:

- Plot confidence, support, lift, number of items using two variables as coordinates and a third as colour;
- Plot the rules as a matrix, with the antecedent (lhs) on the x axis and the consequence on the y axis, reordered with the `seriation` package and/or clustered with the Jaccard distance

$$d(X_i, X_j) = 1 - \frac{|X_i \cap X_j|}{|X_i \cup X_j|};$$

- Graphs (with itemsets, or items, as nodes);
- Parallel plots, etc.

Time-frequency dynamics of the biofuels-fuels-food system
L. Vacha et al. (2012)

To study *one* time series x , one can look at its continuous wavelet transform

$$W_x(u, s) = \int x(t) \frac{1}{\sqrt{s}} \bar{\psi}\left(\frac{t-u}{s}\right) dt.$$

To study two time series x, y , one can look at the cross-wavelet transform

$$W_{xy}(u, s) = W_x(u, s) \overline{W_y(u, s)}$$

or, separately at the *wavelet coherence coefficient* (“time and frequency correlation”)

$$R^2(u, s) = \frac{|S[s^{-1} W_{xy}(u, s)]|^2}{S[s^{-1} |W_x(u, s)|^2] S[s^{-1} |W_y(u, s)|^2]}$$

where S is some (2-dimensional) smoothing operator, and the wavelet phase

$$\phi_{xy}(u, s) = \arg W_{xy}(u, s).$$

GillespieSSA: Implementing the stochastic simulation algorithm in R
M. Pineda-Krch (2008)

Population dynamics (predator-prey, SIR, etc.) are often modeled as coupled ordinary differential equations (ODE), but this is only valid for large populations. For small populations, a stochastic model is needed – *stochastic simulation algorithms* (SSA) are a generalization of the simulation of a Poisson process, modeling the state changes (alive/dead, susceptible/infected/resistant) of the individuals; they can be exact or (faster) approximate.

GrassmannOptim: an R package for Grassmann manifold optimization
K.P. Adragano et al. (2012)

Gradient-based optimization can be used on manifolds other than \mathbf{R}^n – for instance, many constrained optimization problems looking for a set of orthogonal vectors are unconstrained optimization problems on the Grassmannian (e.g., the first eigenvectors, independent components, or other dimension reduction problems).

Graphical models with R: Tutorial
S. Højsgaard (2012)

The `gRbase`, `gRain`, `gRim` packages provide functions to build, use, fit graphical models (or discrete variables, *i.e.*, log-linear models).

***An improved algorithm
for matching large graphs***
L.P. Cordella et al.

To find an isomorphism between two graphs, grow an isomorphism between subgraphs (adding edges outgoing from the subgraph, or incoming if there are no outgoing edges, or other vertices if there are no edges to/from the subgraph), backtracking when needed. The VF2 algorithm is implemented, for instance, in Python's NetworkX module.

McKay's canonical graph labeling algorithm
S.G. Hartke and A.J. Radcliffe

To test if two graphs are isomorphic, one could try all possible permutations, or the permutations (of the vertices) that preserve some vertex invariant, such as the degree (or the centrality, etc.). The degree is too coarse an invariant, but it can be propagated: first, partition the edges by degree; then, for each node, look at the number of edges to each part of the partition – this gives a finer partition; iterate until the partition stabilizes (equitable partition). When you reach an equitable partition, split by removing an element (non-deterministically) from one of the parts (deterministically). The smallest leaf of the resulting search tree (the partitions are ordered) is a canonical isomorph of the graph.

The algorithm is implemented in *nauty*.

It is not known if the graph isomorphism is NP-complete.

Clustering with qualitative information
M. Charikar et al. (2004)

Here are some algorithms to cluster the vertices of a graph whose edges are labeled “agree” or “disagree”:

- Take a binary decision variable x_{ij} for each edge, with $x_{ij} = 0$ if i and j are in the same cluster; the condition $x_{ij} = x_{jk} = 0 \implies x_{ik} = 0$ can be written $x_{ik} \leq x_{ij} + x_{jk}$; minimize

$$\sum_{i,j \text{ agree}} x_{ij} + \sum_{i,j \text{ disagree}} (1 - x_{ij});$$

truncate the solution of the linear relaxation at $2/3$ and $1/3$, one cluster at a time;

- Associate a vector v_i to each vertex: maximizing

$$\sum_{i,j \text{ agree}} v_i v_j + \sum_{i,j \text{ disagree}} (1 - v_i v_j)$$

subject to $v_i v_i = 1$ and $v_i v_j \geq 0$ is a semi-definite relaxation of the previous problem.

***On the identifiability
of the post-nonlinear causal model***
K. Zhang and A. Hyvärinen (2009)

The PNL (post-nonlinear) model $x_2 = f_2(f_1(x_1) + e_1)$ is not identifiable in general (e.g., in the Gaussian case)

but is, in many special cases (e.g., if f_1 is not invertible). It is not directly useable beyond two variables: first use conditional independence tests to find the Markov equivalence class of directed acyclic graphs (DAG), then use the PNL model two variables at a time, to identify the unresolved causal relations.

***Dynamic decentralized any-time
hierarchical clustering***
H.V.D. Parunak et al.

Ant clustering (each ant picks up items and drops them when it is close to similar items) can be seen as a distributed variant of k -means. It can be generalized to produce a hierarchical clustering: start with a random tree (with the data in the leaves) and move nodes (if they are too dissimilar to their siblings) or merge them (if their children are very dissimilar). (This is just a local search algorithm, but an easy-to-parallelize one.)

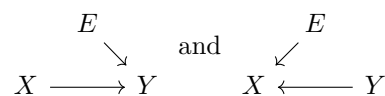
***Parable: a parallel random partition based
hierarchical clustering algorithm
for the MapReduce framework***
S. Wang and H. Dutta

To cluster a large dataset on Hadoop:

- Split the data at random and compute a hierarchical clustering on each node;
- Choose one of the trees as a template;
- Align all the other trees to it: swap two children of the root if doing so makes them more similar to the two children of the root of the template, continue recursively with the other nodes;
- Output the resulting tree: the shape is that of the template, but the leaves contain *sets* of observations rather than individual observations.

***Probabilistic latent variable models
for distinguishing between cause and effect***
J.M. Mooij

The direction of a causal relation between two random variables X and Y can be inferred by comparing the two models



(E is not observed) using Bayesian model selection.

***A quick and gentle guide
to constraint logic programming via ECLPSe***
A. Niederliński (2011)

The book starts with a clear description of the differences between constraint logic programming (CLP) and its ancestor, Prolog: they are syntactically similar, they explore the same tree, but CLP prunes it more efficiently. Prolog only gives up a branch (a partial instantiation, or grounding, of the decision variables)

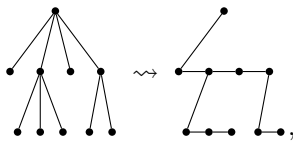
when a constraint (involving ground variables) is violated. CLP keeps track of the domain of each variable (usually, as an interval of integers – but it could also be an arbitrary finite set of integers) and, after each instantiation, propagates the constraints – for instance, if you know that $x + y = z$, $x, y \in \llbracket 0, 3 \rrbracket$, $z \in \llbracket 5, 9 \rrbracket$, you can deduce that $x, y \in \llbracket 2, 3 \rrbracket$, $z \in \llbracket 5, 6 \rrbracket$ – this prunes a lot of branches. In addition, CLP offers more choice for the search strategy: branch-and-bound (Prolog is designed for satisfaction problems and requires dirty, non-declarative tricks), variable choice heuristic (e.g., most constrained), value choice heuristic (e.g., minimum, maximum, middle, random), random restarts, timeout, etc. CLP also differs from operations research (OR), which is limited to continuous and binary variables.

The rest of the book is a long list of examples for the ECLPSe open source CLP system, but most of the language features are used without any explanation.

```
= < > =< >= is ; , not => <= :- \==
#= #< #> #=< #>= :: .. &=: ~ ! -> =\=
$= $< $> $=< $>= fail retractall assert
labeling findall minimize bb_min search
indomain ic_symbolic:indomain
ground suspend delete
```

Similarity evaluation on tree-structured data R. Yang et al. (2005)

The edit distance between trees (number of relabeling, delete and insert operations needed to transform one into the other) can be computed with dynamic programming. Alternatively, one can use the string edit distance between the preorder (or post-order) traversal sequences, or various histogram distances (height, degree, label). Here is another histogram: transform the tree into a binary tree



make it complete, and consider the histogram (empirical distribution) of the binary branches



(they can be seen as 3-grams for binary trees).

OpenTURNS reference guide

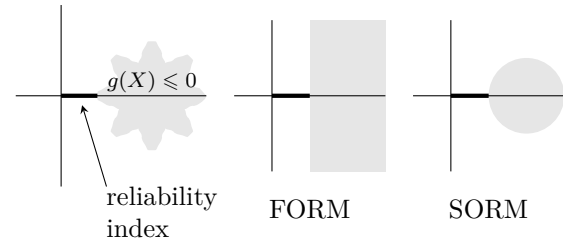
OpenTURNS is a C++ *reliability* library, often used from Python, *i.e.*, a list of methods used by engineers to propagate uncertainty in their computations and identify the main sources of those uncertainties.

First, the input variables can be described, as a joint probability distribution, using empirical cumulative

distribution functions (ecdf), kernel smoothing, correlation, rank correlation, common parametric distributions, copulas, mixtures (via the Fourier transform), classical or Bayesian estimators, tests and plots for goodness of fit, etc.

Then, one propagates uncertainty: either as $[\min, \max]$ intervals, analytically, or by solving optimization problems, or by simulations with quasi-random numbers (low discrepancy sequences, or Latin squares (design of experiments) – Latin squares are fixed-length multivariate low discrepancy sequences); or as location-dispersion pairs, or quantiles, estimated by Monte carlo simulations or Taylor expansions.

To approximate the probability of some event $[g(X) \leq 0]$, transform X to make it rotationally invariant (isoprobabilistic (Nataf, Rosenblatt) transformation – if $X \sim N(\mu, C'C)$, just use $X \mapsto C^{-1}(X - \mu)$) and approximate the event $[g(X) \leq 0]$ with a half-plane (FORM: first order reliability method) or a ball (SORM). The probability can also be approximated with Monte Carlo simulations.



Finally, the importance of each input variable on the uncertainty of the output can be estimated, for instance, if $Z = h(X)$, with a Taylor expansion

$$\begin{aligned} \text{Var } Z &\approx \nabla h(\mu_X) \text{Var } X \nabla h(\mu_X)' \\ &= \sum_i \nabla_i h(\mu_X) \text{Var } X \nabla h(\mu_X)' \end{aligned}$$

or, simply, with $\text{Cor}(Z, X_i)$, $\text{lm}(Z \sim X_i)$, $\text{pCor}(Z, X_i)$, rank correlation, $\partial \text{RI} / \partial \theta$, etc.

Instead of the Taylor expansion, given

$$\begin{array}{ccc} \mathbf{R}^n & \xrightarrow{h} & \mathbf{R}^m \\ & \nwarrow X \quad \nearrow Y=h(X) & \\ & \Omega & \end{array}$$

one can use the *functional chaos expansion*: the expansion of h over a basis of orthogonal polynomials (“polynomial chaos”) wrt f_X (only keep the “first” coefficients – since there are many indices, you can tweak the notion of “first”, e.g., using the L^p pseudonorm, for $p < 1$, of the indices).

LOF: Identifying density-based local outliers M.M. Breunig et al. (2000)

One can use the ideas behind density-based clustering (DBSCAN) to define the degree of outlier-ness of a

point p ,

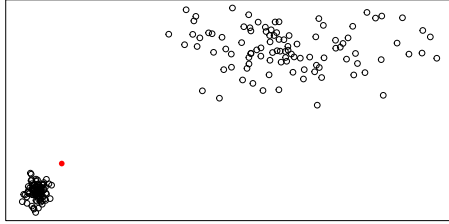
$$\text{LOF}_k(p) = \frac{1}{k} \sum_{q \in N_k(p)} \frac{\ell_k(q)}{\ell_k(p)}$$

$$\ell_k(p) = \frac{1}{k} \sum_{q \in N_k(p)} d_k(p, q) \quad \text{local reachability density}$$

$$d_k(p, q) = \text{Max}\{d_k(q), d(p, q)\} \quad \text{reachability distance}$$

$$d_k(p) = \text{distance to the } k\text{th nearest neighbour.}$$

Being local, the method still works if the data has clusters of different densities. Fraud detection is the most prominent application of outlier detection.



A scalable and efficient outlier detection strategy for categorical data A. Koufakou et al. (2007)

Here are a few ideas to detect outliers in categorical data:

- Greedy algorithm to identify observations whose removal reduces the entropy the most;
- Observations containing few frequent item sets (FIS)

$$\text{FPOF}(x) \propto \sum_{\substack{F \in \text{FIS} \\ F \subset x}} \text{Support}(x)$$

$$\text{FIS} = \{I : |\text{Support } I| > n\}$$

- Observations containing many infrequent itemsets

$$\text{Otle}(x) = \sum_{\substack{I \subset x \\ |\text{Support } I| \leq n}} \frac{1}{|I|}$$

- Observations containing rare values for most variables (*i.e.*, consider the variables separately)

$$\text{AVF}(x) = \frac{1}{m} \sum_i f(x_i)$$

where $f(x_i)$ is the proportion of points in which the i th variable has value x_i .

Frequent pattern growth (FP-growth) algorithm. An introduction F. Verhein (2008)

The *apriori* algorithm for frequent item set (FIS) mining

$$L_1 = \{\text{frequent items}\}$$

$$= \{\{x\} : \text{Support } \{x\} \geq n\}$$

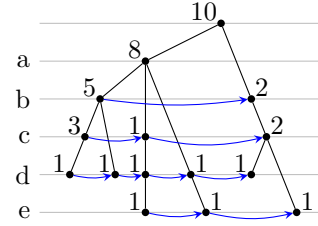
$$C_{k+1} = \{a \cup \{x\}, a \in L_k, \{x\} \in L_1, x \notin a\}$$

$$L_{k+1} = \{c \in C_{k+1} : \text{Support } c \geq n\}$$

is expensive (double loops to compute the candidates C_k and the frequent k -item sets L_k).

The FP-growth algorithm processes the data into an *FP-tree*:

- Compute the support of each item;
- For each transaction, sort the items by decreasing support;
- Build the corresponding *prefix tree*, storing the support of each prefix in the nodes, with links between identical items.

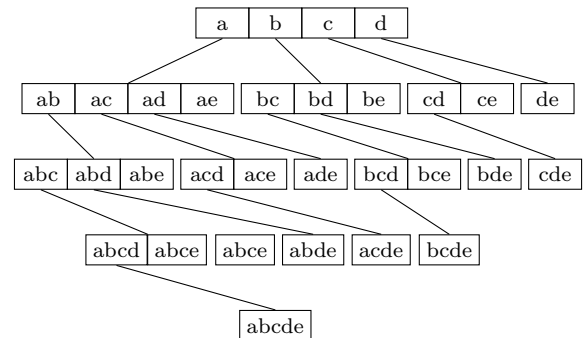


Using the FP-tree to extract frequent itemsets is trickier:

- Start with the least frequent item;
- If it is sufficiently frequent, consider the subtree it defines, update the counts for the conditional FP-tree and discard the leaves (the item itself);
- Process this conditional FP-tree recursively;
- Discard the item and iterate with the next least frequent item, until the tree is empty.

Efficient implementations of Apriori and Eclat C. Borgelt (2003)

The frequent itemset (FIS) mining algorithms *Apriori* and *Eclat* both explore the same prefix tree



breadth-first and depth-first. Eclat can be implemented by storing the transactions as rows in a sparse bitmatrix and by using the columns to compute the intersections.

Simple algorithms for frequent itemset mining
C. Borgelt (2010)

The split-and-merge algorithm for frequent itemset mining preprocesses the data:

- Compute the item frequencies;
- Remove the items below the desired threshold;
- Sort the items in each transaction by increasing frequency;
- Aggregate identical transactions (keep track of their count).

The dataset is then processed recursively (divide and conquer):

- Take the least frequent item;
- Recursively process the itemsets that do not contain it, by removing it from the transactions and reaggregating them (some may become equal);
- Recursively process the transactions that contain it: discard the other transactions, and remove the item from the remaining transactions.

A comprehensive assessment of methods for de novo reverse engineering of genome-scale regulatory networks
V. Nerendra et al. (2010)

Algorithms to identify (undirected) causal relations include:

- Relevance networks: add an edge between variables with a significant mutual information

$$MI(x, y) = \iint p(x, y) \log \frac{p(x, y)}{p(x)p(y)} dx dy$$

(use a kernel estimator for the densities, and change the order of the nested loops to avoid recomputing the 1-dimensional densities $p(x)$, $p(y)$ each time);

- Add an edge between variables with a mutual information above some threshold;
- Prune an MI-derived graph by removing the edge with the lowest mutual information in each triangle (Arcane);
- Methods based on higher order mutual information

$$H(X) = \int -p(x) \log p(x) dx$$

$$H(X, Y) = \iint -p(x, y) \log p(x, y) dx dy$$

$$H(X, Y, Z) = \iiint -p(x, y, z) \log p(x, y, z) dx dy dz$$

$$H(X; Y, Z) = H(X) + H(Y, Z) - H(X, Y, Z)$$

to test if $Y \rightarrow X \leftarrow Z$;

- Add an edge between variables with a significant correlation;
- Add an edge between variables with a significant partial correlation; the partial correlations can be computed from the concentration matrix (the inverse of the variance);
- The graphical lasso, *i.e.*, and L^1 -penalized estimator of the concentration matrix;

- Limited partial correlation (*qp*-graphs);
- Hierarchical clustering;
- Local methods.

Fast calculation of pairwise mutual information for gene regulatory network reconstruction
P. Qiu et al.

To compute the mutual information matrix

$$MI_{k\ell} = \iint p_{k\ell}(x, y) \log \frac{p_{k\ell}(x, y)}{p_k(x)p_\ell(y)} dx dy$$

one often uses a kernel estimator of the probability densities, which gives something like $MI_{k\ell} = \sum_i \log \sum_j f_{ijk\ell}$. This is often implemented with nested loops (for $k \dots$ for $l \dots$ for $i \dots$ for $j \dots$), but one can change the order of the loops (to i, j, k, ℓ) and factor out some of the computations (bits of the kernel estimators of $p_k(x)$, $p_\ell(x)$, which were recomputed each time).

A robust procedure for Gaussian graphical model search from microarray data with p larger than n
R. Castelo and A. Roverato (2006)

Relevance networks infer the structure of a graphical model by testing if $\text{Cor}(X_i, X_j) = 0$, but this ignores confounders, *e.g.*, if $B \leftarrow A \rightarrow C$, then $\text{Cor}(B, C) \neq 0$. Instead, one can look at the conditional independences, which can be tested with partial correlations

$$\text{Cor}(X_i, X_j \mid X_k, k \neq i, j)$$

or, equivalently, the zero pattern of the concentration matrix (the inverse of the variance). When it cannot be estimated, *e.g.*, if $p \gg n$, one can use the *limited partial order correlations* instead,

$$\text{Cor}(X_i, X_j \mid X_{k_1}, \dots, X_{k_q}).$$

The *non-rejection rate* is the proportion of subsets $\{k_1, \dots, k_q\}$ such that

$$H_0 : \text{Cor}(X_i, X_j \mid X_{k_1}, \dots, X_{k_q}) = 0$$

is not rejected (using the T test for zero regression coefficients) – if q is large, use a random subset of the $\binom{p}{q+2}$ subsets. The *qp-procedure* produces the graph whose edges have a non-rejection rate below some threshold. To choose q and assess its adequateness, look at the histogram of the non-rejection rate and the *qp-clique-plot*:

```
plot( max_clique_size ~ threshold | q,
      cex = number_of_edges )
abline( h = min(p,n) )
```

(you want enough edges to capture all the information, but a small maximum clique size).

**Local causal and Markov blanket induction
for causal discovery
and feature selection for classification**
C.F. Aliferis et al. (2010)

Review of a few algorithms to infer *local causality* in a graphical model, viz computing the parents and children (PC) of a node $X \in \mathbf{V}$ and its *Markov blanket* (MB) – a minimal element of

$$\{\mathbf{A} \subset \mathbf{V} \setminus \{X\} : \forall Y \in \mathbf{A} \setminus \{x\} \quad X \perp\!\!\!\perp Y \mid \mathbf{A}\}.$$

For a *faithful graph* (all dependence and independence relations come from the graph),

$$\text{PC}(X) = \{Y \in \mathbf{V} : \forall \mathbf{Z} \subset \mathbf{V} \setminus \{X, Y\} \quad X \not\perp\!\!\!\perp Y \mid \mathbf{Z}\}$$

and $\text{MB}(X)$ is uniquely defined and contains parents, children and childrens' parents (spouses) of X .

By stitching the $\text{PC}(X)$ for all X , one can estimate the whole (global) graphical model.

Local causality learning can also be used for *feature selection*.

Hash kernels for structured data
Q. Shi et al. (2009)

The idea behing the count-min sketch (use a hash table, with several hash functions, and ignore collisions) can be used to approximate scalar products in high-dimensional spaces (e.g., after the “kernel trick”) or to compress sparse feature vectors (in text processing)

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \approx \langle \bar{\phi}(x), \bar{\phi}(y) \rangle$$

$$\bar{\phi}_j(x) = \sum_{i : h(i)=j} \phi_i(x).$$

If the features are duplicated, the drop in performance is not that bad. Hashing can be combined with random sampling – for instance, if the features are the number of subgraphs of size k in each isomorphism class. This idea is implemented in Vowpal Wabbit.

**New methods for separating causes
from effects in genomics data**
A. Statnikov et al. (2012)

Finding the direction of a causal link from observational data alone (no intervention analysis) looks hopeless: in terms of graphical models, $A \rightarrow B$ and $A \leftarrow B$ are in the same Markov equivalence class. However, comparing the complexities of both models, *i.e.*, of the decompositions of $P(A, B)$ into $P(A)P(B|A)$ or $P(B)P(A|B)$, or checking if the conditional distribution is constant, *i.e.*, $B|A \perp\!\!\!\perp A$, can sometimes give the answer – many algorithms have been proposed.

Also check the “Cause-effects pairs” Kaggle competition.

**A linear non-Gaussian acyclic model
for causal discovery**
S. Shimizu et al. (2006)

Variables x_1, \dots, x_n , related by cause/consequence relations described by a directed acyclic graph (DAG), and observed with a non-Gaussian, additive noise ε , can be modeled as $x = Bx + \varepsilon$, where B is strict lower triangular. But since $x = (I - B)^{-1}\varepsilon$, independent component analysis (ICA) can recover $B = I - \text{ICA}(x)$, up to a permutation of the rows and columns. The resulting DAG can be pruned with statistical tests for the strength of the causal relations.

**Distinguishing causes from effects using
non-linear acyclic causal models**
K. Zhang and A. Hyvärinen (2007)

The LiNGAM algorithm (ICA-based causality discovery) can be extended to allow for non-linearities

$$x_{k+1} = f_{k+1}(g_{k+1}(x_1, \dots, x_k) + \varepsilon_{k+1}),$$

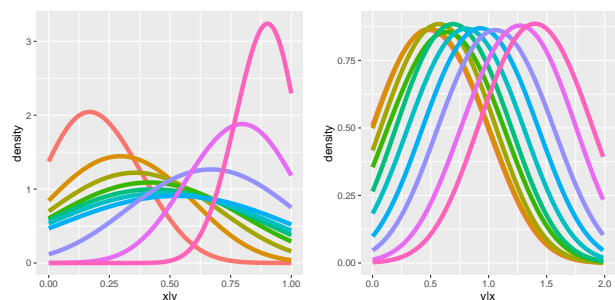
which can be reformulated as

$$x = f(Bs)$$

with B strict lower triangular. In the case of two variables, one can find g_1 and g_2 so that x_1 be as independent as possible from $g_2(x_2) - g_1(x_1)$, e.g., by minimizing their mutual information with a multilayer perceptron (MLP) and using a statistical test to check the independence.

**Nonlinear causal discovery
with additive noise models**
P.O. Hoyer et al.

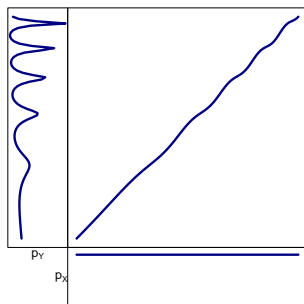
Given a non-linear noise with *additive* non-Gaussian noise $y = f(x) + \varepsilon$, plot the density of $y|x$ for several values of x : it should be the same curve, up to translations. This property is unlikely to hold for $x|y$ and strengthens the belief that $x \rightarrow y$.



The actual test checks that $x \not\perp\!\!\!\perp y$ and $\text{res}(y \sim f(x)) \perp\!\!\!\perp y$, where $y \sim f(x)$ is a non-linear regression and $\text{res}(\cdot)$ its residuals.

Inferring deterministic causal relations
P. Daniušis et al. (2012)

Causal inference often relies on the assumption that the noise is additive and non-Gaussian. This breaks down in the case of a deterministic relation $Y = f(X)$, especially if f is invertible. However, one can expect $X \perp\!\!\!\perp f$ and $Y \not\perp\!\!\!\perp f$, where $\perp\!\!\!\perp$ denotes *algorithmic independence* – the shortest description of $p(x, y)$ is a separate description of $p(x)$ and f .



Given a family of distributions \mathcal{E} well-suited to model a probability density p , one can measure the complexity of p as

$$D(p|\mathcal{E}) = \min_{q \in \mathcal{E}} D(p||q),$$

where

$$D(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx$$

is the Kullback-Leibler divergence. The information-geometric causal inference (IGCI) method computes

$$C_{X \rightarrow Y} = D(p_X || \mathcal{E}_X) - D(p_Y || \mathcal{E}_Y)$$

and concludes that $X \rightarrow Y$ if $C_{X \rightarrow Y} < 0$. If $\mathcal{E}_X = \mathcal{E}_Y$ only contains the uniform distribution $U(0, 1)$,

$$\begin{aligned} C_{X \rightarrow Y} &= \int_0^1 \log |f'(x)| p(x) dx \\ &\approx \frac{1}{m-1} \sum_{i=1}^{m-1} \log \left| \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \right|, \end{aligned}$$

where $x_1 \leq \dots \leq x_m$.

The architecture of SciDB
M. Stonebraker (2011)

SciDB is a database for scientific data, not unlike a column store, but with arrays instead of columns. It can be queried with a SQL-like language (AQL) or a more procedural one (AFL). In addition to relational operators (join, etc.), it provides scientific operations (matrix multiplication, singular value decomposition (SVD), regression, machine learning). Data is stored in *overlapping* chunks, because many operations (on time series, images, geographical data, etc.) require neighbouring values. It also supports versioning (the data is never deleted, but stored to optimize queries with current data).

Latent Dirichlet allocation: towards a deeper understanding
C. Reed (2012)

A *topic* is a probability distribution on a collection of words. Latent Dirichlet allocation (LDA) is similar to k -means, but with bags of words instead of vectors in \mathbf{R}^n : it models a “mixture” of topics, one step of the algorithm assigning each text to the nearest topic, the next refining the topics by “averaging” the texts associated with it.

The R package metaLik for likelihood inference in meta-analysis
A. Guolo and C. Varin (2012)

Meta-analysis (combining the result of many studies, without access to the raw data of each), boils down to a seemingly simple random-effects model: estimate β in

$$\begin{aligned} Y_i &= \beta_i + e_i \\ \beta_i &= \beta + \varepsilon_i \\ e_i &\sim N(0, \sigma_i^2) \quad \text{precision of study } i \\ \varepsilon_i &\sim N(0, \tau^2) \quad \text{heterogeneity of the studies.} \end{aligned}$$

Unfortunately, most statistical procedures (tests, confidence intervals, etc.) rely on asymptotic results: the small size of the sample (the number of studies aggregated) makes them invalid. In this context, higher-order expansions (of the log-likelihood) are preferable.

Non-parametric kernel distribution function estimation with kerdie: an R package for bandwidth choice and applications
A. Quintela-del-Río and G. Estévez-Pérez (2012)

If you are more interested in the cumulative distribution function (cdf) than the probability distribution function (pdf), e.g., probability of exceedance, mean return period, quantiles, etc., the optimal bandwidth for kernel density estimation is different.

Spherical k-means clustering
K. Hornik et al. (2012)

Spherical k -means, often used in text clustering, refers to k -means for the cosine dissimilarity, *i.e.*, after projecting the data on the sphere, $\min_{p,c} \sum_i 1 - \cos(x_i, p_{c(i)})$, *i.e.*,

$$\min_{M,p} \sum_{ij} \mu_{ij} (1 - \cos(x_i, p_j)),$$

with $\mu_{ij} \in \{0, 1\}$ and $M\mathbf{1} = \mathbf{1}$. In the fixed point algorithm, local improvements are possible (change the membership of a single observation; this also changes the prototypes p). Extended spherical k -means minimize

$$\sum_{ij} w_{ij} \mu_{ij}^m (1 - \cos(x_i, p_j)),$$

with $\mu_{ij} \in [0, 1]$ and $M\mathbf{1} = \mathbf{1}$ (m and w are given).

ClustOfVar:
an R package for clustering of variables
M. Chavent et al. (2012)

PCAMIX is a generalization of principal component analysis (PCA) that allows both quantitative and qualitative variables. The first component is

$$\operatorname{Argmax}_{\mathbf{u} \in \mathbf{R}^n} \sum_{j \text{ quantitative variable}} \operatorname{Cor}^2(\mathbf{u}, \mathbf{x}_j) + \sum_{j \text{ qualitative variable}} \operatorname{Corr}^2(\mathbf{u}, \mathbf{y}_j)$$

where $\operatorname{Corr}(\mathbf{u}, \mathbf{y})$ is the *correlation ratio*, i.e., the proportion of the variance of \mathbf{u} explained by \mathbf{y} . It can be computed from the singular value decomposition (SVD). It can also be used to cluster the variables, by considering

$$\operatorname{Argmax}_{\substack{\mathbf{u}_1, \dots, \mathbf{u}_k \in \mathbf{R}^n \\ \mathcal{C}_1, \dots, \mathcal{C}_k \text{ partition} \\ \text{of the variables}}} \sum_{\ell=1}^k \sum_{j \in \mathcal{C}_\ell \cap \text{Quant}} \operatorname{Cor}(\mathbf{u}, \mathbf{x}_j) + \sum_{j \in \mathcal{C}_\ell \cap \text{Qual}} \operatorname{Cor}(\mathbf{u}, \mathbf{y}_j).$$

This can be approximated by hierarchical clustering of the variables, or a k -means-like algorithm.

The devil is in the tails: actuarial mathematics and the subprime mortgage crisis
C. Donnelly and P. Embrechts (2010)

The default times, needed to price CDOs, are often modeled with a 1-parameter Gaussian copula and (say) exponential margins. This is inadequate: it implies asymptotic independence – one should use another copula or, at least, stress the model with other copulas.

R and data mining: examples and case studies
Y. Zhao (2013)

Examples (working code with little or no explanations of what is computed) of data mining algorithms in R. The paper version also contains case studies. The topics include:

- Decision trees: `party::ctree`, `rpart`, `randomForest`;
- Regression: `lm`, `glm`, `nls`;
- Clustering: `kmeans`; `cluster::pam`, `cluster::clara`, `fpc::pamk`; `hclust`; `fpc::dbscan`;
- Outlier detection: `boxplot.stats`; `DMwR::lofactor`, `Rlof::lof`; `dbscan`, `kmeans`, time series models, `extremevalues`, `mvoutlier`, `outliers`;
- Time series: `decompose`, `stl`, `timsac::decomp`, `ast::tsr`; `arima`; `dwt`; clustering with `dwt::dwtDist`; classification after transforming the time series into features, e.g., with `dwt`, or with k -nearest-neighbours (`RANN::nn`);
- Association rules: `arules::apriori`, `arules::eclat`, `arulesViz`;
- Text mining: `twitterR::userTimeline`; `tm::vectorSource`, `Corpus`, `tm_map`, `getTransformations`, `TermDocumentMatrix`, `find*`; `wordcloud`; `textcat` (n -grams), `lda`, `topicmodels`;
- Social network analysis: `igraph`, `sna`

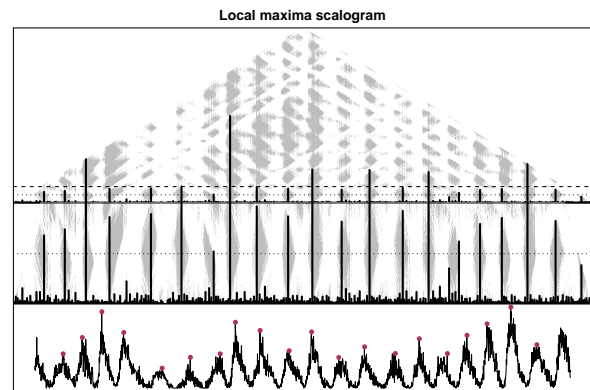
There is no mention of the forecast and caret packages.

An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals
F. Scholkmann et al. (2012)

The *local maxima scalogram* (LMS)

$$m_{k,i} = \mathbf{1}_{x_i < x_{i-k}, x_i > x_{i+k}}$$

may help detect peaks in a signal.



Graph databases
I. Robinson et al. (2013)

Relational databases are cumbersome and inefficient when dealing with graphs (self-joins, recursive joins), and nullable columns complicate queries even further. NoSQL databases can fake foreign keys, but all the work has to be done by the application. Graph databases use index-free adjacency (the relations are not stored in a global, separate index, but locally, bidirectionally, at each node), often separate graph structure from property data (not unlike column stores), and provide efficient graph-theoretic operations (depth-first search, breadth-first search, shortest path (Dijkstra), A^* , etc.). They can be used for predictive modeling, e.g., identify missing links (*triadic closure*: if $A-B$ and $A-C$, then $B-C$ is likely) or important links (a *local bridge* is a relation that leads to a different part of the network).

Applications include social data (find colleagues with similar interests, colleagues of colleagues interested in a given topic, etc.), email forensics, recommendations, geographical data (parcel delivery and shortest path queries, R-trees), master data management, network and data center management (machines, applications, etc.), access control (dealing with complex organization structures and product hierarchies requires recursive joins in SQL), bioinformatics (protein networks).

Data modeling is very similar to entity-relation (ER) modeling: use a node for each entity (and for most relations: only encode simple relations as relationships), avoid attributes on relationships (prefer fine-grained relationships). Time can be stored in linked lists (previous/next) or date nodes (or, even, a timeline tree:

each day points to its month, each month points to its year). Optimizing a graph database just means adding direct links for relations that would otherwise require several links.

Neo4j provides several APIs: a core API (in Java, with `Node` and `Relationship` objects), a traversal API (more declarative), and Cypher (a SQL-like language – the book has a lot of examples); SPARQL (for RDF) and Gremlin are other query languages. It can be embedded (either persisted on disk, or in-memory), used as a server (REST, JSON – the server can be extended: use JAX-RS annotations to indicate to which URIs a class/method responds), supports transactions, recovery, master-slave replication.

Data-intensive text processing with MapReduce

J. Lin and C. Dyer (2010)

1. With the help of infrastructures such as MapReduce, the “unreasonable effectiveness of data” has made data-intensive scientific discovery the “fourth paradigm of science” (after theory, experiment and simulations).

Contrary to most books on MapReduce or its implementation Hadoop, which hide actual contents (if any) under complicated installation instructions (this is Java) and boilerplate code (this is Java), this one focuses on MapReduce *algorithm* design – no actual code, just clear Python-like pseudo-code.

2. MapReduce assumes that failures are common, moves processing closer to the data and processes data sequentially (no random access). It can be seen as a cloud analogue of the map and fold primitives of functional programming.

$$\begin{aligned}\text{map: } (k_1, v_1) &\longrightarrow [(k_2, v_2)] \\ \text{reduce: } (k_2, [v_2]) &\longrightarrow [(k_3, v_3)]\end{aligned}$$

The Hadoop infrastructure works as follows:

- The input data is stored in a distributed file system (HDFS), often serialized (ProtocolBuffers, Avro, Thrift); the files are stored on *data nodes* (each file is replicated three times); the metadata is on the *name node*.
- The *mapper* receives the data, as key-value pairs, and emits more key-value pairs.
- The *combinator* (optional) pre-aggregates the data: for instance, when counting words, emitting pairs $(w, 1)$ is inefficient


```
def mapper(id, terms):
    for term in terms:
        emit term, 1
def reducer(term, counts):
    emit term, sum(counts)
```

 and the counts can be pre-aggregated.
- The *partitionner* (optional) groups the data, usually by key but, for some applications, you may want to use only part of the key.

- The framework shuffles and sorts the data (transparently);
- The *reducer* receives (key,list) pairs and emits more key-value pairs.
- The *jobtracker* oversees the MapReduce jobs (on the job submission node), the *tasktrackers*, on each data node, run the code.

3. Here are a few MapReduce design patterns:

- *In-mapper recombiner*: while it is possible to pre-aggregate the data in a combiner, it is often more efficient to do it in the mapper – not in its `map` method (called for each key-value pair), but in its `close` method (called when the node has finished processing all the data). You may need to flush the accumulator if it gets too large. For some operations, e.g., the mean, the combiner and the reducer do different things.


```
def mapper(t,x):
    emit t, (x,1)
def combiner(t, xcs):
    xs, cs = zip(*xcs) # unzip
    emit t, (sum(xs), sum(cs))
def reducer(t, xcs):
    xs, cs = zip(*xcs) # unzip
    emit t, sum(xs)/sum(cs)
```
- *Pairs*: to compute co-occurrences, use pairs of words as keys.
- *Stripes*: to compute co-occurrences, use one word as key, and a hash table as value.
- *Order inversion*: if you need the result of the aggregation before it is ready, e.g., to compute relative frequencies, duplicate the data (e.g., one copy for the word count, another for the word pair count), make sure the intermediate result (word count) is computed first (set the sort order), and partition the data accordingly.


```
emit (w1,*):1
emit (w1,w2):1
emit (w1,w3):1
partition on w1
```
- *Value-to-key conversion*: when the reducer must produce sorted data, ask the framework to sort it before, by putting the value to sort in the key.
- *Reduce-side join*: for a 1-to-1 join of \mathcal{T} and \mathcal{S} , emit $k : (t, T)$ and $k : (s, S)$ (where s and t are the primary keys of \mathcal{S} and \mathcal{T} , k the column to join on, and T, S the other columns), for a 1-to-many join, emit $(k, s) : S$ and $(k, t) : T$, define a custom partitioner, ensure that the values of the small table come first and cache them.
- *Map-side join*: if the data has already been partitioned on disk, e.g., as a result of a previous MapReduce process, map over the larger set after reading the corresponding chunk of the smaller one in the mapper – no reducer is needed.
- *Memory-backed join*: put the smaller dataset into memory (or some distributed key-value store: Memcached, etc.) in every mapper – if it does not fit into memory, partition it into n pieces and perform n in-memory joins.

4. An *inverted index* maps words to sorted lists of document ids (there may be some data (“payload”) with the id: term frequency, position, style (title or not), HTML link, linguistic function (POS, type of entity: place, person), etc.). For efficient set operations (intersection, union), the list should be sorted: do not emit `term:(id,freq)`, but `(term,id):freq` to have the framework sort the ids (you need a custom partitioner to ensure that all the messages for a given term are sent to the same reducer).

The volume of data is huge: the process can benefit from compression.

- Only store the differences (*d*-gaps) between document ids – since they are sorted, the numbers are smaller.
- Variable length *integer coding*: set the 8th bit to zero as long as the number is not finished.
- Group varInt: group the numbers by 4 and prefix each group with a byte indicating (on 2 bits) the length of each number.
- Simple-9: in each 32-bit word, use 4 bits to indicate how the remaining 28 bits are split into equal-sized parts.
- Unary code: encode n as $n - 1$ 1s and one 0.
- γ -code: encode n as $\lfloor \log_2 n \rfloor$ 1s, a 0, and the binary encoding of the number, without the leading 1.
- δ -code: similar to the γ code, but the first part is not unary-encoded but γ -encoded.
- Golomb code: choose b , encode $q = \lfloor (n - 1)/b \rfloor$ in unary, encode the remainder $r = n - 1 - qb$ in truncated binary (the smaller numbers are encoded on k bits, the others on $k + 1$ – unassigned k -bit codes followed by 0 or 1).

MapReduce is a poor solution for document retrieval, but you could partition the documents, and cache the most-requested results.

5. Many sequential graph algorithms rely on some global data structure (e.g., a priority queue): they are not straightforward to implement in MapReduce. For instance, breadth-first search (Dijkstra’s algorithm) can be implemented as a sequence of MapReduce iterations: store the estimated distance from the source in each node, process each node in each iteration, stop when the distances no longer change – there are as many iterations as the longest shortest path (you also need to keep track of the adjacency list, e.g., by emitting it as well).

```
def mapper(id,d):
    for a in neigh(id):
        emit a, d+w
def reducer(id,ds):
    emit id, min(ds)
```

PageRank (a random walk on the graph, with teleportation, computing a score (equilibrium probability) for each node), HITS (random walk on a bipartite graph of hubs and authorities, computing two scores for each node on the initial graph – the graph is usually

query-dependent: pages containing the search terms and their neighbours), SALSA (idem) can be computed in the same way, each iteration spreading the probability mass (but pay attention to dangling nodes). Combiners can improve performance, if you partition the data with some heuristic (sort by zipcode, school, language, domain name, etc.). To avoid underflow, use log-probabilities: they can be added as

$$a \oplus b = \begin{cases} b + \log 1p(e^{a-b}) & \text{if } a < b \\ a + \log 1p(e^{b-a}) & \text{if } a \geq b. \end{cases}$$

6. If a statistical model contains parameters θ (unobserved, to be estimated), hidden variables y (unobserved, but we do not care much about their values), and data x (observed), maximum likelihood estimators

$$(\hat{\theta}, \hat{y}) = \underset{(\theta, y)}{\operatorname{Argmax}} P(X = \mathbf{x}, Y = \mathbf{y}; \theta)$$

give a very noisy estimator of θ : prefer the *marginal likelihood* estimator (i.e., integrate y out):

$$\begin{aligned} \hat{\theta} &= \underset{\theta}{\operatorname{Argmax}} P(X = \mathbf{x}; \theta) \\ &= \underset{\theta}{\operatorname{Argmax}} \prod_i \sum_y P(X = x_i, Y = y_i; \theta). \end{aligned}$$

The *expectation-maximization* (EM) algorithm is a hill-climbing algorithm that attempts to maximize the marginal log-likelihood:

$$\theta_{n+1} \leftarrow \underset{\theta}{\operatorname{Argmax}} \sum_{x,y} P(x, y; \theta_n) \log P(x, y; \theta).$$

This single formula is often, confusingly, presented as two steps:

- Estimation of the probability distribution $X, Y \mid \theta = \theta_n$ in the E step (it is a probability distribution: in the discrete case, it is a set of probabilities);
- Computation of the expected log-probability

$$\sum_{x,y} P(x, y; \theta_n) \log P(x, y; \theta)$$

(it is a function of θ : if it is explicitly computed, it is also part of the E step);

- Maximization, in the M step.

Given a hidden Markov Model (HMM), the problems of computing the probability that the model generated the data (used for clustering or outlier detection) and computing the most probable sequence of hidden states that could generate the observed data can be solved with dynamic programming (the forward algorithm computes the probability that the process is in state q at time t ; the Viterbi algorithm computes the probability of the most probable sequence of states leading to state q at time t): they can be implemented with MapReduce, by processing one column of the dynamic programming table (one time t) in each iteration; each mapper receives a small part of the data; each reducer computes a cell in the dynamic programming table. Similarly, estimating the parameters of a HMM (forward-backward algorithm) can be implemented with EM.

HMM can be used, for instance, to align texts in different languages:

$$\begin{aligned} P(\text{target, alignment} \mid \text{source}) \\ &= \prod P(t_{i+1} \mid t_i) \times \prod P(t_i \mid s_{a_i}) \\ &= \text{language model} \times \text{translation model}. \end{aligned}$$

MapReduce can also be used for parallel but non-data-intensive tasks.

The book also mentioned, with few details, other technologies in the MapReduce/Hadoop ecosystem: Mahout (machine learning), HBase (tables in HDFS), Cassandra (key-value store), Giraph (bulk-synchronous parallel (BSP) framework), GPU programming (e.g., for streaming data), Pig, Hive, Hadoop (data processing).

Financial risk modelling and portfolio optimization with R **B. Pfaff (2013)**

Each chapter is structured in the same way: terse presentation of a few mathematical notions (consider it as a mere check-list), list of relevant R packages (not unlike the CRAN task views), and (more interesting) examples.

The topics covered are

- Risk measures: VaR, ES, mVaR, mES, coherent measures;
- Efficient frontier;
- Distributions to model returns: generalized lambda distribution (GLD), GHD (ghyp, lmomco::pargld, fBasics::gldFit);
- Extreme value theory: block maxima, peaks over thresholds, exceedance declustering (evir::gev, esmev::gev.fit, fExtremes::gevFit, ismev::rlarg.fit, fExtremes::mrlPlot, gpdFit, deCluster);
- Volatility: GARCH models (fGarch::garchFit, rugarch);
- Dependence: correlation, rank correlation, Kendall's τ , lower tail dependence, copula GARCH model (fit a GARCH model to individual time series, and then a copula to the (joint) residuals), mixture of copulas (QRM::fit.tcopula, QRM::rcopula.t, copula::dcopula);
- Robust estimation: M-estimator, MM-estimator, MVE, MCD, S-estimator, SDE, OGK estimator (rrcov::Cov*);
- Robust optimization: scenario-based, box or elliptic uncertainty set (FRAPO::Socp, Rsocp);
- Diversification: risk contribution, diversification ratio, defined with the volatility or some downside risk measure such as the tail dependence coefficient (FRAPO::P*, FRAPO::dr, FRAPO::cr, Portfolio::Analytics::optimize.portfolio);
- Portfolio optimization with VaR, ES or drawdown in the objective or the constraints (fPortfolio::minRiskPortfolio, FRAPO::P*);

- Time series models: ARMA, VAR, VECM, SVAR, SVEC (urca, vars);
- Tactical asset allocation: Black-Litterman, copula opinion pollomg, entropy polling (TTR, fTrading, BLCOP).

Lectures on modern convex optimization **A. Ben-Tal and A. Nemirovski (2012)**

Convex programming is too general to be amenable to practical, efficient methods, but many special cases are – linear, conic quadratic, semi-definite programming are all examples of *conic programming*.

1. To find a lower bound of

$$x^* = \underset{x}{\text{Min}} \{f(x) : \forall i \, g_i(x) \geq b_i\},$$

consider a linear combination of the constraints

$$\sum_i y_i g_i(x) \geq \sum_i y_i b_i$$

(with $y \geq 0$). If $f(x) \geq \sum_i y_i g_i(x)$, then $\sum y_i b_i$ is a lower bound. The best such lower bound is

$$\underset{y \geq 0}{\text{Max}} \left\{ \sum_i y_i b_i : \forall x \, f(x) \geq \sum_i y_i g_i(x) \right\}.$$

This is the *dual problem*; its optimal value is less than that of the primal (weak duality). If f and g are linear, set x to $+1$ and -1 : the inequalities become equalities, and the dual of

$$\underset{x}{\text{Min}} \{c'x : Ax \geq b\}$$

is

$$\underset{y}{\text{Max}} \{b'y : y \geq 0, A'y = c\}.$$

For linear problems, there is a *strong duality*: the primal has an optimal solution iff the dual has, and the optimal values are the same – but there are two ways in which a problem can fail to have a solution: it can be infeasible, or unbounded; if the primal is unbounded, then the dual is infeasible, but the converse is false: both can be infeasible.

Applications of linear programming include compressed sensing (ℓ^1 penalty as an approximation of a (non-convex) ℓ^0 one), support vector machines (SVM), discrete-time linear dynamic systems.

Replacing the linear objective function by an arbitrary (convex) function is not the only way of generalizing linear programming into non-linear programming – one can keep the objective function linear but replace the constraints $Ax - b \geq 0$ with $Ax - b \succ_K 0$, where \succ_K denotes the partial order induced by a closed, pointed convex cone with non-empty interior K . In the dual problem, $y \geq 0$ gets replaced by $y \succ_{K^*} 0$, i.e., $y \in K^*$, the dual cone. This is **conic programming**.

To formulate the dual of

$$\underset{x}{\text{Min}} \{c'x : Ax \succ_K b\},$$

we are looking for a lower bound on the objective

$$c'x \geq \langle \lambda, Ax \rangle \geq \langle \lambda, b \rangle$$

for some λ . But an arbitrary λ does not preserve inequalities: the dual of K is the set of λ that always do:

$$K^* = \{\lambda : \forall x \succ_K 0 \langle \lambda, x \rangle \geq 0\}.$$

For the dual, consider

$$\text{Max}_{\lambda} \{\langle \lambda, b \rangle : \lambda \succ_{K^*} 0, \forall x \langle c, x \rangle \geq \langle \lambda, Ax \rangle\}.$$

The last condition can be written as

$$\forall x \langle c, x \rangle \geq \langle A^* \lambda, x \rangle.$$

By setting x to a basis vector and its opposite, we see that this is equivalent to $c = A^* \lambda$. The dual is therefore:

$$\text{Max}_{\lambda} \{\langle \lambda, b \rangle : \lambda \succ_{K^*} 0, A^* \lambda = c\}.$$

The primal can be formulated in the same way (intersection of a cone and an affine subspace) by adding a variable $y = Ax - b$. More compactly:

$$\text{primal} : \text{Min}_{\lambda} \{\langle d, y \rangle : y \in \mathcal{L} - b, y \succ_K 0\}$$

$$\text{dual} : \text{Max}_{\lambda} \{\langle b, \lambda \rangle : \lambda \in \mathcal{L}^\perp + d, \lambda \succ_{K^*} 0\}.$$

Strong duality holds if the primal is strictly feasible, *i.e.*, $\exists x \ Ax - b \succ_K 0$ (strict inequality), *i.e.*, $\overset{\circ}{K} \cap (\mathcal{L} - b) \neq \emptyset$. (Without strict feasibility, anything can happen: primal solvable but dual infeasible, etc.)

2. The Lorentz cone

$$L^m = \left\{ x \in \mathbf{R}^m : x_m \geq \sqrt{x_1^2 + \dots + x_{m-1}^2} \right\};$$

is self-dual. A **second order cone program** (SOCP) is

$$\text{Min}_x \{c'x : Ax - b \in L^{m_1} \times \dots \times L^{m_k}\}.$$

The condition $Ax - b \in L^m$ can be written,

$$\begin{pmatrix} Dx \\ p'x \end{pmatrix} - \begin{pmatrix} d \\ q \end{pmatrix} \in L^m,$$

where

$$\begin{pmatrix} D & d \\ p' & q \end{pmatrix} = \begin{pmatrix} A & b \end{pmatrix},$$

i.e., $p'x - q \geq \|Dx - d\|_2$.

The dual of

$$\text{Min}_x \{c'x : \forall i \ \|D_i x - d_i\|_2 \leq p'_i q - q_i\}$$

is

$$\begin{aligned} &\text{Find} \\ &\text{To maximize} \quad \sum_i^{\mu, \nu} \mu'_i d_i + \nu_i q_i \\ &\text{Such that} \quad \sum_i D'_i \mu_i + \nu_i p_i = c \\ &\quad \forall i \ \|\mu_i\|_2 \leq \nu_i. \end{aligned}$$

Optimization problems rarely arise in this form: they have to be somehow transformed. For the objective, replace “minimize $f(x)$ ” with “minimize t such that $t \geq f(x)$ ”. For the constraints, there is a very, very long list of functions or constructions that can be used in SOCPs; here are a few of them.

– The Euclidian norm: $\|x\|_2 \leq t$;

– Its square:

$$\begin{aligned} \|x\|_2^2 \leq t &\iff x'x + \frac{1}{4}(t+1)^2 \leq \frac{1}{4}(t-1)^2 \\ &\iff \left\| \begin{pmatrix} x \\ \frac{1}{2}(t-1) \end{pmatrix} \right\|_2 \leq \frac{t+1}{2} \end{aligned}$$

– Convex quadratic forms (*i.e.*, quadratic programming):

$$\begin{aligned} x' C' C x + q' x + r &\leq t \\ \iff \left\| \begin{pmatrix} Cx \\ \frac{1}{2}(t + q'x - r) \end{pmatrix} \right\|_2 &\leq \frac{1}{2}(t - q'x - r); \end{aligned}$$

– Many (rational) powers;

– The L^p norm;

– etc.

Robustifying a linear program with elliptic (or, more generally, conic-quadratic representable) constraints gives a SOCP. For instance, the robustification of the constraint $ax - b \geq 0$ is

$$\text{Min}_{a^*, b^*} \left\{ a^* x - b^* : \left\| \begin{pmatrix} a^* - a \\ b^* - b \end{pmatrix} \right\|_2 \leq \varepsilon \right\} \geq 0$$

i.e.,

$$ax - b + \text{Min}_{u, v} \left\{ \begin{pmatrix} x \\ -1 \end{pmatrix}' \begin{pmatrix} u \\ v \end{pmatrix} : \left\| \begin{pmatrix} u \\ v \end{pmatrix} \right\|_2 \leq \varepsilon \right\} \geq 0$$

i.e.,

$$ax - b - \varepsilon \left\| \begin{pmatrix} x \\ -1 \end{pmatrix} \right\|_2 \geq 1.$$

(To robustify the objective $c'x$, replace it with a new variable t , add the constraint $c'x \leq t$, and robustify it.) The robustified problem can be used to assess the stability of the objective, the solution, the feasibility status of a linear problem.

3. Semi-definite programming (SDP) is another special case of cone programming, where the (self-dual) cone is that of positive semi-definite symmetric matrices. One can assume there is only one inequality – otherwise, put them in a single block-diagonal matrix. It is a generalization of conic quadratic programming:

$$\begin{pmatrix} x \\ t \end{pmatrix} \in L^k \iff \begin{pmatrix} t I_{k-1} & x \\ x' & t \end{pmatrix} \succ 0.$$

In addition, SDP can use the largest eigenvalue (or singular value), the sum of the k largest eigenvalues (or singular values), the spectral norm, (some rational powers of) the determinant, non-negative polynomials, trigonometric polynomials, etc. Applications include the stability of dynamical systems (eigenvalues) and robust cone programming.

Semi-definite programming provides *relaxations* of combinatorial problems, finer than LP relaxations, as follows. Formulate the problem as a quadratic program

with quadratic constraints (e.g., $x \in \{0,1\}$ is equivalent to $x^2 - x = 0$ or, since we prefer inequalities, $x^2 - x \leq 0$, $x - x^2 \leq 0$).

$$x^* = \underset{x}{\operatorname{Argmin}}\{f(x) : g(x) \leq 0\}$$

Consider the dual problem, *i.e.*, transform the constraints to penalties, with coefficients to be determined,

$$f_\lambda(x) = f(x) + \lambda g(x), \lambda \geq 0;$$

this provides a lower bound on $f(x^*)$

$$\zeta^* = \inf_x f_\lambda(x) \leq f(x^*).$$

But, since $f_\lambda(x)$ is a quadratic form, $\forall x \zeta \leq f_\lambda(x)$ means that $f_\lambda(\cdot) - \zeta \geq 0$. Finding the best bound is therefore a semi-definite program:

$$(\zeta^*, \lambda^*) = \underset{\zeta, \lambda}{\operatorname{Argmax}}\{\zeta : f_\lambda(\cdot) - \zeta \geq 0, \lambda \geq 0\}.$$

The SDP relaxation of a quadratically-constrained quadratic program can also be formulated as follows: replace inhomogeneous quadratic forms

$$g(x) = x'Ax + 2b'x + c$$

with homogeneous ones

$$G(x, t) = x'Ax + 2tb'x + ct^2 = \begin{pmatrix} t \\ x \end{pmatrix}' \begin{pmatrix} c & b' \\ b & A \end{pmatrix} \begin{pmatrix} t \\ x \end{pmatrix},$$

then, replace the quadratic forms $x'Gx$ with $\operatorname{tr}(GX)$. The relaxation corresponds to the embedding

$$\begin{cases} \mathbf{R}^n \rightarrow S_+^{n+1} \\ x \mapsto \begin{pmatrix} 1 \\ x \end{pmatrix} \begin{pmatrix} 1 \\ x \end{pmatrix}' \end{cases}.$$

Applications of SDP relaxation include graph problems (Shannon capacity, maxcut), other combinatorial problems, chance constraints.

Finding the best inner ellipsoidal approximation of a polytope defined by inequalities (or, more generally, of an intersection of ellipsoids) is a SDP.

Finding the best outer ellipsoidal approximation of a polytope defined as the convex hull of its vertices (or, more generally, of a union of ellipsoids) is a SDP.

4. Convex programming is a polynomial problem: the *ellipsoid method* needs $O(n^2 \log \varepsilon^{-1})$ steps (and each steps needs $O(n^2)$ operations) and goes as follows. Start with an ellipsoid containing the optimal set. If its center is not feasible, there exists a separating hyperplane between the center and the (convex) feasible set: use it to cut the ellipsoid. If it is feasible, the (sub) gradient of the objective defines a hyperplane, on one side of which the optimal set lies: use it to cut the ellipsoid. In both cases, take an outer approximation of the cut ellipsoid and iterate.

Interior point methods are interior penalty methods, *i.e.*, to solve

$$\underset{x}{\operatorname{Min}}\{c'x : x \in \mathcal{X}\},$$

they solve a sequence of unconstrained problems

$$x_*(t) = \underset{x}{\operatorname{Argmin}} tc'x + K(x)$$

where K is a *barrier function* (if $x_n \rightarrow x$ and $x \in \partial\mathcal{X}$, then $K(x_n) \rightarrow +\infty$) and t increases, $(x_*(t))_{t \geq 0}$ traces the *central path* (each is solved with the Newton method, which converges quadratically if sufficiently close to the solution).

With conic programming, a clever choice of the barrier function

$$\begin{aligned} x \geq 0 &\rightsquigarrow -\log x \\ X \succ 0 &\rightsquigarrow -\log \det X \\ x \in L^k &\rightsquigarrow -\log(x_k^2 - x_1^2 - \dots - x_{k-1}^2) \end{aligned}$$

and $t_{n+1} = 1.1t_n$, a single Newton step suffices for each value of t and the algorithm converges in $O(\log \varepsilon^{-1})$ steps. It is easier to monitor convergence and stay close to the central path if one traces both the primal and dual central paths; they are related by the augmented *complementary slackness* relation

$$\begin{aligned} x^*(t) &= -t^{-1} \nabla K(s^*(t)) \\ s^*(t) &= -t^{-1} \nabla K(x^*(t)) \end{aligned}$$

(since the cone is self-dual, the barrier is the same for both problems).

5. Interior point methods are polynomial, but for *large-scale problems* $O(n^3)$ or $O(n^2)$ will not do: if we want the algorithm to terminate in a reasonable amount of time, we can afford to evaluate the objective, its gradient, but nothing fancier.

There are information-theoretic bounds on the number of steps needed to achieve a given precision ε , e.g., $O(\log n / \varepsilon^2)$ as $n \rightarrow \infty$ for ball constraints (and much worse for box constraints) – it is almost independent of the dimension.

Mirror descent is a generalization of the projected gradient

$$x_{n+1} \leftarrow \operatorname{Prox}_{x_n}(\gamma_t f'(x_t)),$$

where one replaces the norm, distance and gradient to better suit the problem. For instance,

- Over an L^2 ball, $\|\cdot\|_2$, $\omega(x) = \frac{1}{2}x'x$ and the proxy mapping

$$\begin{aligned} \operatorname{prox}_x \xi &= \underset{y \in X}{\operatorname{Argmin}} \omega(y) + \langle \xi - \omega'(x), y \rangle \\ &= \underset{y \in X}{\operatorname{Argmin}} \|x - \xi - y\|_2 \end{aligned}$$

is the projection of $x - \xi$ on X ;

- Over the simplex Δ_n , use $\|\cdot\|_1$ and $\omega(x) = \sum x_i \log x_i$;
- Over the *spectahedron* $\{x \in S^n : x \geq 0, \operatorname{tr} x \leq 1\}$, use $\|\lambda(x)\|_1$ and $\omega(x) = \sum_i \lambda_i(x) \log \lambda_i(x)$.

A convex-concave saddle point problem (e.g., equilibrium, in a zero-sum game)

$$\text{Min}_{x \in X} \text{Max}_{y \in Y} \phi(x, y),$$

where ϕ is convex in x , concave in y , can be solved with mirror descent by considering the vector field (ϕ_x, ϕ_y) instead of the gradient.

There are a few variants: stochastic mirror descent if the gradient is noisy (the expected error converges in the same way), bundle mirror descent (momentum, to address plateau problems), etc.

Most non-smooth optimization problems are of the form

$$\text{Min}_{x \in X} \text{Max}_{y \in Y} \phi(x, y)$$

and can be smoothed as (Nesterov smoothing)

$$\text{Min}_{x \in X} \text{Max}_{y \in Y} \phi(x, y) + d(y).$$

The *mirror prox* algorithm (a variant of saddle point mirror descent) can be used to solve those problems. Convergence, for a non-smooth optimization problem, using only first-order information, is at best $O(1/\sqrt{rt})$, but it can be brought to $O(1/t)$ after such a transformation. Examples of non-smooth functions that can be used in this way include:

- The L^p norm, $\|x\|_p = \text{Max}_{\|y\|_q \leq 1} \langle y, x \rangle$.
- The L^p norm of the singular vectors of a matrix;
- The L^p -norm of the positive part,

$$\|x^+\|_p = \text{Max}_{\substack{\|y\|_q \leq 1 \\ y \geq 0}} \langle y, x \rangle$$

- The maximum entry of a vector, or the sum of the k largest entries

$$s_k(x) = \text{Max}_{y \in \Delta_{n,k}} \langle y, x \rangle$$

where $\Delta_{n,k} = [\mathbf{1}'y = k, y \geq 0]$;

- The sum of the k largest eigenvalues

$$S_k(x) = \text{Max}_{\substack{y \geq 0 \\ \text{tr } y = k}} \text{tr}(yx).$$

Maximum likelihood estimation of a multi-dimensional log-concave density M. Cule et al. (2010)

The log-concave maximum likelihood density estimator is well-defined, and asymptotically finds the log-concave density that minimizes the Kullback-Leibler divergence with the true density. Its support is the convex hull of the sample data (log-concave distributions have thin tails).

The estimator can be used for data visualization (e.g., checking for the presence of elliptic contours), classification (estimate the density of each class), clustering (fit a mixture of log-concave distributions with the EM algorithm, as you would a mixture of Gaussians), monte Carlo estimation of functionals of the

density (probabilities, moments, expectations, entropy, etc.), mixing detection (compare the log-concave estimator with a mixture of log-concave densities, or with $f(x) \propto \exp(\phi(x)_c \|x\|^2)$, with ϕ concave and fixes values $c > 0$ (this also accounts for fat tails), or with a permutation test comparing the data with samples from the estimated distribution).

Contrary to kernel-based estimators, this does not require the choice of a bandwidth matrix (problematic in high dimensions).

The density is found by minimizing

$$\sigma(y_1, \dots, y_n) = -\frac{1}{n} \sum y_i + \int_C \exp h_y(x) dx$$

where $y_i = \log \hat{f}(x_i)$ are the log-densities evaluated at the data points x_i , h_y is the smallest concave function with $\forall i \ h_y(x_i) \geq y_i$, the first term is the (negated) log-likelihood, the second term can be thought of as a Lagrangian term (we want the density to integrate to 1), C is the convex hull of the points (the support of the density). Since the concave function h_y is affine on each triangle of a triangulation of the points (obtained as a side result of the QuickHull algorithm), the integral is easy to compute. The objective function is convex, but non-differentiable: one could use Newton's algorithm with a subgradient, but convergence would be slow (linear or worse, versus quadratic for differentiable functions), or *Shor's r algorithm* (space dilation in the direction of the difference of two consecutive gradients – empirically faster)

There is an R implementation in the `LogConcDEAD` package (in dimension 1, also check `logconden`).

Efficient rank reduction of correlation matrices

I. Grubišić and R. Pietersz (2005)

A constrained optimization problem

$$\text{Max}_{x \in \mathbf{R}^n} \{f(x) : g(x) = 0\}$$

can be formulated and solved as an unconstrained one if the feasible set $M = [g(x) = 0] \subset \mathbf{R}^n$ is a manifold M on which you can explicitly compute the gradient and (for the Newton and conjugate gradient methods) the hessian: the gradient comes from the isomorphism between $T_y M$ and $T_y^* M$ induced by the Riemannian structure,

$$F_y(u) = \langle (\text{grad } F)_y, u \rangle_{T_y M},$$

the hessian from the Levi-Civita connection and the updates to the solution (“moving in some direction”) are given by parallel transport along geodesics.

In the case of rank-constrained correlation matrices, the feasible set is not a manifold (it is a stratified manifold, each stratum corresponding to a value for the rank), but it can be described as the quotient of a manifold (the “Cholesky manifold”) by the orthogonal

group. The resulting algorithm is equivalent to optimization using a parametrization with spherical coordinates.

Sparse inverse covariance selection via alternating linearization methods
K. Scheinberg et al. (2010)

There are many algorithms for sparse inverse covariance selection (SICS), *i.e.*, sparse estimators of the inverse covariance matrix corresponding to sparse graphical models ($\Sigma_{ij}^{-1} = 0$ iff $X_i \perp\!\!\!\perp X_j | X_{k, k \neq i, j}$):

$$\operatorname{Max}_{X \succ 0} \log \det X - \langle X, \Sigma \rangle + \lambda \|X\|_1.$$

The alternating linearization method solves the optimization problem

$$\operatorname{Min}_x f(x) + g(x)$$

with f and g convex, by rewriting the problem as

$$\operatorname{Min}_{x, y} \{f(x) + g(y) : x = y\}$$

and separately updating x and y . It can be adapted to solve the SICS problem.

Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data
O. Banerjee et al. (2006)

The sparse inverse covariance matrix

$$\operatorname{Argmax}_{X \succ 0} \log \det X - \operatorname{tr} SX - \rho \|X\|_1,$$

where S is the sample variance matrix, can be estimated by solving the dual problem, which is of the form

$$\operatorname{Max}_W \{\log \det W : \|W - S\|_\infty \leq \lambda\}.$$

It can be estimated one coordinate at a time (block-coordinate descent)

$$w_{12} \leftarrow \operatorname{Argmin}_y \{y' W_{11}^{-1} y : \|y - s_{12}\|_\infty \leq \rho\}.$$

Sparse inverse covariance estimation with the graphical lasso
J. Friedman et al. (2007)

This problem can in turn be solved though its dual

$$\operatorname{Min}_\beta \frac{1}{2} \left\| W_{11}^{1/2} \beta - b \right\|^2 + \rho \|\beta\|_1,$$

which is a lasso problem. The **glasso** package provides an implementation.

Covariance selection and estimation via penalized normal likelihood
N. Liu et al.

The coefficients of the Cholesky matrix can be interpreted as regression coefficients – penalized (lasso, ridge) regression gives a penalized Cholesky matrix, and a penalized variance matrix.

Covariance selection for non-chordal graphs via chordal embedding
J. Dahl et al.

The sparsity pattern of a variance matrix forms a chordal graph. One can easily estimate the inverse covariance matrix under a chordal sparsity constraint and, with more work, an arbitrary sparsity constraint.

Sparse permutation invariant covariance estimation
A.J. Rothman et al. (2008)

The L^1 -penalized inverse-variance (concentration) estimator

$$\operatorname{Argmin}_{\Omega \succ 0} \operatorname{tr}(\Omega \hat{\Sigma}) - \log |\Omega| + \lambda \|\Omega^-\|_1$$

(where Ω^- are the off-diagonal elements of Ω) can be estimated by parametrizing Ω as $\Omega = T'T$, with T lower-triangular, to ensure positivity, and using a quadratic approximation of the absolute value:

$$|u_{n+1}| \approx \frac{1}{2} \frac{u_{n+1}^2}{|u_n|} + \frac{1}{2} |u_n| \quad \text{or} \quad \frac{1}{2} \frac{u_{n+1}^2}{|u_n| + \varepsilon} + \frac{1}{2} |u_n|.$$

Efficient estimation of covariance selection models
F. Wong et al. (2003)

Bayesian approach for covariance selection (sparse estimation of the inverse covariance), with a Γ prior on the diagonal elements, and a zero-inflated prior on the correlations, estimated with Gibbs sampling, updating one partial correlation at a time (to keep the matrix positive definite).

Shrinkage algorithms for MMSE covariance estimation
Y. Chen et al. (2009)

How to choose the shrinkage coefficient when estimating a covariance matrix (the optimal coefficient depends on the unknown variance matrix and cannot be used, the Ledoit-Wolf coefficient can be improved on).

Identifying small mean-reverting portfolios
A. d'Aspremont (2008)

When looking for cointegrated assets to form mean-reverting portfolios, sparse portfolio are more investable. For instance, one could look for a portfolio whose prices $P_t = S_t x$ maximize the Ornstein-Uhlenbeck mean reversion parameter λ

$$dP_t = \lambda(\bar{P} - P_t)dt + \sigma dZ_t.$$

As a proxy for the mean-reversion, one can use the Box-Tiao *predictability*

$$\nu = \frac{\text{Var}_{t-1}[P_{t-1}]}{\text{Var}_{t-1}[P_t]} = \frac{\text{Var}[P_{t-1}]}{\text{Var}[P_t|P_{t-1}]}$$

(maximizing mean-reversion corresponds to minimizing predictability, *i.e.*, momentum). If the asset prices follow a VAR(1) process $S_t = S_{t-1}A + Z_t$, then $P_t = S_t x = S_{t-1}Ax + Z_t x$ and

$$\nu(x) = \frac{x' A' \Gamma A x}{x' \Gamma x}$$

where $\Gamma = \text{Var } S_t$ – it is a generalized eigenvalue problem.

To estimate the matrices A and Γ , one can use penalized methods (covariance selection for Γ , lasso for A).

The sparse generalized eigenvalue problem

$$\text{Max}_x \left\{ \frac{x' Ax}{x' Bx}, \|x\|_0, \|x\| = 1 \right\}$$

can be approximately solved greedily (progressively increase the number of non-zero coefficients, assuming that those sets of indices are increasing) or by semi-definite relaxation.

An open-source implementation of the critical line algorithm for portfolio optimization
D.H. Bailey and M. López de Prado (2013)

When people compute the efficient frontier

$$\begin{array}{ll} \text{Find} & w \\ \text{To minimize} & w' \Sigma w \\ \text{and maximize} & w' \mu \\ \text{Such that} & w' \mathbf{1} = 1 \\ & l \leq w \leq u \end{array}$$

they often solve a series of optimization problems, each with a different target return, and interpolate. This is imprecise and computationally wasteful.

The *turning points* on the efficient frontier are efficient portfolios such that nearby efficient portfolios contain different assets – points, on the efficient frontier, at which an asset enters or leaves the portfolio. Between two turning points, the efficient portfolios are the solution of an unconstrained problem (only equality constraints: $w' \mu = \mu_{\text{target}}$ and $w' \mathbf{1} = 1$) on a subset of the assets, for which an analytic expression can be derived (via Lagrange multipliers); they are also convex

combinations of the turning points (2-fund theorem). The efficient frontier can be computed by starting with the maximum return portfolio and moving down, from turning point to turning point.

The authors provide an implementation in Python.

Loglog counting of large cardinalities
M. Durand and P. Flajolet (2003)

To approximately count the number of different elements in a stream of words (words in a text, IP addresses in an intrusion detection system, values in a column in a database for query optimization, etc.) one can:

- Distribute the hashed (randomized) values in N buckets, in a bitmap, and count the occupied buckets (there are adaptive and hierarchical variants);
- Keep a proportion $p \ll 1$ of the data (by looking at the first bits of the hashed value) and count it, exactly;
- Look at the maximum position $\rho(x)$ of the first non-zero bit in the hashed data: $\log_2 N \approx \text{Max } \rho(x)$.

The log-log algorithm uses this idea, after splitting the data into m buckets, and averages the estimates.



HyperLogLog: the analysis of a near-optimal cardinality estimation algorithm
P. Flajolet et al. (2007)

In the loglog counting algorithm, one can replace the arithmetic mean of the \hat{N}_i with the harmonic mean of the $2^{\hat{N}_i}$.

Firefly algorithm, stochastic test functions and design optimisation
X.S. Yang (2010)

The firefly algorithm is similar to particle swarm optimization (PSO), but each particle moves randomly (no momentum) and is more attracted by nearby fitter particles (it is not attracted by less fit particles at all). If the local minima are hardly distinguishable, the particles can end up in several of them.

Empirical mode decomposition of financial data
K. Drakakis (2008)

An intrinsic mode function (IMF) is a continuous function with positive maxima and negative minima, *i.e.*, a function that oscillates (e.g.,  or ). A function f can be decomposed into an IMF (empirical mode decomposition, EMD) as follows (sifting):

- Interpolate the local maxima (upper envelope) and local minima (lower envelope) of f ;
- Average them;
- Subtract this average from the function;
- Iterate until this difference is an IMF;
- Subtract the IMF from f and start again to decompose the residuals.

**Recent mathematical development
on empirical mode decomposition**
Y. Xu and H. Zhang

The first intrinsic model function (IMF) of the empirical mode decomposition (EMD) is intuitively similar to the Hilbert transform (HHT, Hilbert-Huang transform):

$$Hf(t) = \text{pv} \int_{\mathbf{R}} \frac{f(s)}{t-s} ds$$

$$Af = f + iHf$$

$$Af(t) = \rho(t)e^{i\theta(t)}$$

where $\rho(t)$ is the instantaneous amplitude, $\theta(t)$ the phase, $\theta'(t)$ the frequency.

This technical article tries to describe which functions can be obtained as IMF.

In R, check the EMD and hht packages.

MCMC Using Hamiltonian dynamics
R.M. Neal (2011)

To sample from a probability distribution $P(q) \propto \exp -U(q)$, the Hamiltonian Monte Carlo (HMC) method adds another variable p (momentum) and considers the Hamiltonian system $H(q, p) = U(q) + K(p)$, with $K(p) = \frac{1}{2}p'Mp$; its dynamics are

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i}$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i}.$$

To be numerically stable, the discretization of this system should remain reversible, volume-preserving, symplectic. The Euler method does not, but the *leapfrog* method (start with a half-step for momentum, and then use full steps for both position and momentum so that the updates are staggered) does. The HMC algorithm goes as follows: start with a state (q, p) , sample p from $P(p) \propto \exp -K(p)$ (Gaussian), make L leapfrog steps of size ϵ , and use the new state as a Metropolis proposal. HMC avoids the random-walk-like behaviour of the Metropolis algorithm, especially when the dimension is high or the variables correlated; fine-tuning the mass matrix M can improve things even further. Choosing ϵ and L is difficult, though.

**The No-U-Turn Sampler:
Adaptively setting path lengths
in Hamiltonian Monte Carlo**
M.D. Hoffman and A. Gelman

The leapfrog simulation in HMC sampling is usually done a fixed number of times L . To avoid having to specify L , one can try to move in one direction, as long as $\|q_{\text{new}} - q_{\text{old}}\|$ increases, *i.e.*, as long as

$$\frac{1}{2} \frac{d}{dt} \|q_{\text{new}} - q_{\text{old}}\|^2 = (q_{\text{new}} - q_{\text{old}}) \cdot p_{\text{new}} \geq 0.$$

But this is not time-reversible: we would not be sampling from the right distribution. The NUTS algorithm fixes that problem by moving $s_n 2^n$ steps ahead, for increasing values of n , with s_n a random sign (the technical details are more complicated than that).

Stan modeling language
M.D. Hoffman and A. Gelman (2012)

Stan is a Bayesian sampler, not unlike Bugs or Jags, but it uses Hamiltonian Monte Carlo (HMC) sampling: when a Gibbs sampler converges slowly and generates correlated samples, HMC should perform better. The model is converted into heavily-templated C++ (formal differentiation of the log-likelihood) and compiled (this can take time). Stan can automatically select the tuning parameters: the mass matrix is set to the identity; the number of steps can be chosen with the NUTS algorithm; the step size can be estimated during warm-up. Instead of a sampling statement

```
y[i] ~ normal(mu, sigma)
```

one can explicitly increment the log-probability

```
log__ <- log__ + normal_log(y[i], mu, sigma)
```

This is useful to integrate out discrete parameters.

**PyMC: Bayesian stochastic modeling
in Python**
A. Patil et al. (JSS 2010)

PyMC is a Bayesian sampler, not unlike Bugs or Jags, using Python instead of a more limited DSL (domain-specific language).

```
import numpy as np
from pymc import *
n = 5 * np.ones(4, dtype=int)
x = np.array([-0.86, -0.3, -0.05, 0.73])
alpha = Normal('alpha', mu=0, tau=.01)
beta = Normal('beta', mu=0, tau=.01)
@deterministic
def theta(a=alpha, b=beta):
    return invlogit(a+b*x)
d = Binomial('d', n=n, p=theta,
    value=np.array([0,1,3,5]), observed=True)
M = MCMC(d)
M.isample(iter=10000, burn=1000, thin=10)
M.trace('alpha')[:]
plot(M); M.stats()
g = geweke(M); geweke_plot(g)
raftery_lewis(M)
Matplot.autocorrelation(M)
```

It can also deal with “factor potentials”, *i.e.*, the unnormalized probabilities that appear in undirected graphical models.

Convergence can be assessed with the Geweke test,

$$z = \frac{\hat{\theta}_{\text{begin}} - \hat{\theta}_{\text{end}}}{\sqrt{\text{Var } \theta_{\text{begin}} + \text{Var } \theta_{\text{end}}}},$$

comparing the beginning (say, the n th decile) and the end (say, the last half) of the chain, or the Raftery

and Lewis procedure, that estimates the required burn-in, sequence length and thinning to have a $q = 97.5\%$ quantile, precise at $r = .5\%$, with probability $s = 95\%$,

$$P[|\hat{q} - q| \leq r] \geq s.$$

Goodness of fit can be estimated by sampling from the fitted model and comparing with the real data, e.g., by comparing the discrepancy $\langle (x - \hat{x})^2 \rangle$ (where \hat{x} is the (fitted) expected value of x) of the two datasets.

Five balltree construction algorithms **S.M. Omohundro (1989)**

Balltrees are an alternative to k-d-trees (which split the space in two at each node, orthogonally to one axis) and octrees (which split the space into 2^d quadrants at each node): a binary tree of balls, not necessarily disjoint, with the children included in the parents. They are used with two types of queries: pruning (e.g., find all leaves containing a given query node) and branch&bound (e.g., finding the nearest neighbours).

There are a few algorithms to build them:

- k-d algorithm: choose the axis along which the range of the data is larger, split the balls along the median of their centers;
- Top-down: idem, but choose the dimension and split point to minimize the total volume of the bounding balls;
- Insertion: add a new observation as a sibling of an existing leaf, in the position that increases the volume the least;
- Bottom-up: find the two balls whose bounding ball has the lowest volume, merge them, iterate;
- Improved bottom-up: idem, but each node keeps track (e.g., with another balltree implementation) of the best ball to merge it with, and the resulting volume; the balls are in a priority queue and only need to be updated when they are removed from the queue.

The bottom-up construction gives better results (smaller volume) for continuous distributions, manifolds and hierarchical clusters (e.g., the Cantor set).

An unscented Kalman smoother for volatility extraction: evidence from stock prices and options **J. Li (2011)**

To estimate the volatility from a stochastic volatility model: discretize it, linearize it, and apply a Kalman smoother. The unscented Kalman smoother does not explicitly linearize the model, but uses several points around the current state ($2d + 1$ of them: one in each axis direction) to estimate the mean and variance needed by the Kalman filter.

A short introduction to learning to rank **H. Li (2011)**

In information retrieval (IR, e.g., web search), the computer returns the top 100 answers and somehow ranks

them. Those two steps are often separate (the relevance function used to extract the top 100 is not used to sort the results): one can use machine learning to learn that sorting function.

To evaluate the ranking, use a penalty for misranked results, (logarithmically) higher if the difference is large, (exponentially) smaller if the element is far from the top.

Pointwise methods use standard classification methods (ordinal regression). Pairwise methods learn the function $(x, y) \mapsto x \leq y$ or $(x, y) \mapsto \text{rank } x - \text{rank } y$. Listwise methods learn an \mathbf{R} -valued score function (not uniquely defined) that induces the desired order.

Most of the examples presented are variants of support vector machines (SVM).

What HMMs can do **J. Bilmes (2002)**

Hidden Markov models (HMM) are graphical models, *i.e.*, they can be defined by a set of conditional independence relations between the state S and the emitted message M :

$$\{S_{[t,T]}, M_{[t,T]}\} \perp\!\!\!\perp \{S_{[1,t-2]}, M_{[1,t-2]}\} \mid S_{t-1} \\ M_t \perp\!\!\!\perp \{S_{-t}, M_{-t}\} \mid S_t.$$

In other words:

- Future $\perp\!\!\!\perp$ past \mid present;
- Given the current state, the message is independent of all the other variables.

Gaussian processes in machine learning **C.E. Rasmussen**

A Gaussian process is an infinite family of random variables $(X_t)_{t \in \mathbf{R}}$; the mean and covariance functions

$$m(t) = E[X_t], \quad k(s, t) = \text{Cov}(X_s, X_t)$$

suffice to define it. The joint distribution of X_{t_1}, \dots, X_{t_n} is Gaussian, and the conditional distribution $X_{s_1}, \dots, X_{s_m} \mid X_{t_1}, \dots, X_{t_n}$, obtained in the usual way (Shur complement) is the posterior distribution.

To train a Gaussian process, one can use a hierarchical prior, *i.e.*, posit a parametrization of the mean and covariance functions, e.g.,

$$m(t) = at^2 + bt + c \\ k(s, t) = \sigma_1^2 \exp\left(-\frac{(s-t)^2}{\ell^2}\right) + \sigma_2^2 \delta_{ij}$$

(the $\sigma_2^2 \delta_{ij}$ term accounts for noisy observations) and select the hyperparameters via log-marginal likelihood.

***A dynamic programming
segmentation procedure
for hydrological and environmental time series***
A. Kehagias et al. (2005)

The empirical quality of a time series segmentation can be measured by the squared error of a constant, linear of AR(1) model estimated on each segment; the optimal segmentation with a given number of segments can be efficiently computed ($O(n^2)$) via dynamic programming. The number of segments can be estimated with the Bayesian information criterion (BIC).

If the true segmentation is known, the quality of the result can be assessed by looking at the number of misclassified pairs (i, j) with $|i - j| < N/2$.

***Löwdin orthogonalization
A natural supplement to Gram-Schmidt***
S. Beaver

Contrary to Gram-Schmidt orthogonalization, Löwdin orthogonalization

$$\operatorname{Argmin}_{Q \in O(n)} \|A - Q\|_F$$

is symmetric. It can be computed from the singular value decomposition (SVD) $A = U\Sigma V'$ as $Q = UV'$.

***Statistical outliers and dragon-kings
as Bose-condensed droplets***
V.I. Yukalov and D. Sornette (2012)

Model for power law distribution with endogenous outliers ("dragon-kings", as opposed to exogenous outliers or "black swans"): let $\varepsilon(n)$ be the number of cities with n inhabitants or more, *i.e.*, the rank of a city with n inhabitants; $w(\varepsilon) = ne^{-\beta\varepsilon}$ the attraction of a city with rank ε ; $P[\varepsilon(n) \leq \varepsilon] = aw(\varepsilon)^n$; and a boundary condition $\varepsilon(m) = \text{number of cities (i.e., all cities have } m \text{ or more inhabitants)}$ – the outliers come from this boundary condition.

***Linking agent-based models and stochastic
models of financial markets***
L. Feng et al. (2012)

Evidence from agent-based-models that fat tails and long memory come from herding among technical traders.

***Advances in cointegration and subset
correlation hedging methods***
**M.M. López de Prado and D. Leinweber
(2012)**

Hedging a portfolio P_1 with assets P_2, \dots, P_n means finding w so that the spread

$$S_t = P_{1,t} + \sum_{k \geq 2} w_k P_{k,t}$$

be "small", in the sense that $S_T = 0$, or $S_{T+h} - S_T = 0$, or S_T small, or $S_{T+h} - S_T$ small, or $\text{Var } S_T$ small, or S stationary, etc.

Here are a few heging algorithms:

- Regression (only valid if the intercept is not significant);
- Minimum variance (equivalent to no-intercept regression), or minimum risk, for your preferred measure of risk (VaR, CVaR, etc.);
- Principal components: let V be the first $n - 1$ principal components of $\text{Var } \Delta P$, and choose w so that $V'w = 0$, $w_1 = 1$;
- Find w to minimize the Dickey-Fuller statistic (or the statistic of some other unit root test);
- Maximize the diversification ratio $\frac{\sum w_k \sigma_k}{\sigma_{\Delta S}}$
- Minimize the maximum subset correlation.

The *predictability* of an AR(1) process $P_t = \beta P_{t-1} + \varepsilon_t$ is

$$\lambda_t = \frac{E_{t-1}[P_t^2]}{E[P_t^2]} = 1 - \frac{E[\varepsilon_t^2]}{E[P_t^2]}.$$

If the asset prices P_{it} follow a VAR(1) model, the predictability of a linear combination $w'P_{\cdot,t}$ is defined similarly (after simplification, it is a generalized eigenvalue). The *Box-Tiao method* minimized the predictability of the spread.

***Economic scenarios for an asset and liability
management study of a pension fund***
C.C. Slagmolen

VAR models are often used to generate scenarios for ALM computations; the state variables could be 1-month interest rate, 10-year interest rate, inflation, equity returns and dividend yield; one can impose that some of the coefficients of the model be zero to account for causal relations.

$$z_{t+1} = \nu + Bz_t + \Sigma\xi_{t+1}$$

Those state variables can be used to estimate the value of the assets of a pension fund, but cannot be used, directly, to discount future liabilities: the term structure generated by a VAR model can present arbitrage opportunities. To estimate the term structure, one can first model the deflator (sometimes called stochastic discount factor or pricing kernel) M_{t+1} on $[t, t + 1]$

$$\begin{aligned} -\log M_{t+1} &= \delta_0 + \delta_1 z_t + \frac{1}{2} \lambda_t' \lambda_t + \lambda_t' \xi_{t+1} \\ \lambda_t &= \lambda_0 + \Lambda z_t \end{aligned}$$

(where $\delta_0 + \delta_1 z_t$ is the short rate and λ_t the market price of risk) and assume that the term structure is affine, *i.e.*, that the price, at time t , of a zero-coupon bond maturing at time $t + n$, is $P_t^{(n)} = \exp(A_n + C_n z_t)$. But the stochastic discount factor also gives a price, $P_t^{(n+1)} = E_t[M_{t+1} P_{t+1}^n]$, allowing us to compute A_n and C_n .

The model is very sensitive to the starting state (e.g., low interest rates) and the estimation periods (e.g., presence of a crisis).

***Optimal versus naive diversification:
how inefficient is the 1/N portfolio strategy?***
V. DeMiguel et al. (2007)

Comparison of the 1/N portfolio strategy with alternatives (minimum variance or tangent portfolio from sample variance, bayesian estimators, shrinkage estimators, prior from a CAPM or Fama-French pricing model, etc.): it is not significantly worse. Those alternatives have an edge when there is a lot of data, when the correlation is not invariant under the action of \mathfrak{S}_n , when there are few assets, and/or when the optimal Sharpe ratio is significantly higher than the 1/N one.

In defence of optimization: the fallacy of 1/N
M. Kritzman et al. (2010)

If you have enough data (5 to 50 years) or use reasonable expected returns, optimal portfolios significantly outperform 1/N portfolios.

***Why does an equal-weighted portfolio
outperform value- and
price-weighted portfolios?***
Y. Plyakha et al. (2012)

The equal-weighted portfolio outperforms the value-weighted portfolio, not only because of a different risk exposure (more small caps), but also because of frequent rebalancing.

Adaptive asset allocation
Macquarie Private Wealth

Amateurish (plots with no axes, confusion between correlation and volatility, plots that seem to omit the quantities of interest (performance, etc.) as if there was something to hide, ridiculous numeric precision, etc.) advocacy for risk parity portfolios (keep the contribution to risk constant, lower the leverage when the risk is too high) that ends with a minimum variance portfolio on high-momentum stocks.

Performance attribution for equity portfolios
Y. Lu and D. Kane

The Brinson method decomposes the performance of a portfolio as follows. Let W_j^B (resp. W_j^P) be the weight of sector j in the benchmark (portfolio), R_j^B (resp. R_j^P) the returns sector j in the benchmark (portfolio).

Active return

$$\begin{aligned} &= \text{Portfolio returns} - \text{Benchmark returns} \\ &= W^P R^P - W^B R^B \\ &= (W^P - W^B) R^B + W^B (R^P - R^B) \\ &\quad + (W^P - W^B) (R^P - R^B) \end{aligned}$$

The three terms are called “allocation”, “selection” and “interaction”. This can be generalized to several groupings (e.g., sectors and countries), but the number of interaction terms increases exponentially.

With several periods, you can neither add the returns (the aggregated returns would be wrong) nor compound them (the returns are correct, but the contributions no longer add up to them). Instead, one can use *optimal linking*: write the returns as

$$R_{[0,T]}^P - R_{[0,T]}^B = \sum_t \beta_t (R_t^P - R_t^B)$$

for some judiciously-chosen “linking coefficients” β :

$$\begin{aligned} \beta_t &= A + \alpha_t \\ A &= \frac{(R_{[0,T]}^P - R_{[0,T]}^B)/T}{(1 + R_{[0,T]}^P)^{1/T} - (1 + R_{[0,T]}^B)^{1/T}} \\ \alpha_t &= C(R_t^P - R_t^B) \\ C &= \frac{R_{[0,T]}^P - R_{[0,T]}^B - A \sum_t (R_t^P - R_t^B)}{\sum_t (R_t^P - R_t^B)^2}. \end{aligned}$$

Regression-based analysis can more easily accomodate several groupings or factors (Brinson analysis can be seen as a no-intercept regression), but time-aggregation poses the same problems.

In R, this is implemented in the pa package.

Equity performance attribution methodology
Morningstar (2008)

Since the interaction term, in the Brinson method, is difficult to interpret, it can be incorporated into the last decision: in the (preferred) top-down approach, the contributions are allocation and selection+interaction, in the bottom-up approach, they are selection and allocation+interaction.

For multi-period attribution, one can use a geometric approach:

$$\begin{aligned} \log(1 + R^P) - \log(1 + R^B) &= \\ \log(1 + R_{\text{allocation}}) + \log(1 + R_{\text{selection}}) & \end{aligned}$$

where $R_{\text{allocation}}$ and $R_{\text{selection}}$ are decomposed into sums of contributions.

Long and short positions should be analyzed separately. One can add a “return gap” term, for unexplained returns (intra-period trades, corporate actions, etc.)

***How news affect the trading behaviour
of different categories of investors
in a financial market***
F. Lillo et al. (2012)

Endogenous information (returns, volatility), exogenous information (news – they use the (non-free) General Enquirer; check C. Potts’s lexicons web page for more lists of words) and investor behaviour.

Comparing the performance of FA, DFA and DMA using different synthetic long-range correlated time series
Y.H. Shao et al. (2012)

Empirical comparison of various long-range correlation estimators (Hurst index):

- R/S (rescaled range) analysis;
- FA (fluctuation analysis): $\langle (y_{t+s} - y_t)^2 \rangle^{1/2} \sim s^{-\alpha}$;
- DFA (detrended FA): $\langle (y_{t+s} - g_t)^2 \rangle^{1/2} \sim s^{-\alpha}$, where g_t is a polynomial approximation of y_t on non-overlapping boxes of size s ;
- DMA: idem, with a moving average with window size s .

DFA is good enough.

Numerical evaluation of a generalized Cauchy principal value
A. Nyíri and L. Baranyi (1999)

The Cauchy principal value

$$\text{pv} \int_{x_0-\Delta}^{x_0+\Delta} \frac{f(x)}{h(x) - h(x_0)} dx$$

can be evaluated numerically as $\int_0^\Delta g$, where

$$g(u) = \frac{f(x_0 + u)}{h(x_0 + u) - h(x_0)} + \frac{f(x_0 - u)}{h(x_0 - u) - h(x_0)}$$

$$g(0) = \frac{f'_+ + f'_-}{h'} - f \frac{h''_+ + h''_-}{2h'}.$$

Principal-value integrals by a simple and accurate finite-interval method
W.J. Thompson (1997)

The Cauchy principal value can be approximated numerically, if the interval around the singularity is small, by using the Taylor expansion (only the odd derivatives remain).

$$\text{pv} \int_{x_0-\Delta}^{x_0+\Delta} \frac{f(x)}{x - x_0} dx = 2 \sum_{n \geq 0} \frac{f^{(2n+1)}(x_0) \Delta^{2n+1}}{(2n+1)!(2n+1)}$$

This can be generalized to $\text{pv} \int_{x_0-\Delta}^{x_0+\Delta} \frac{f(x)}{(x - x_0)^p} dx$.

Isoelastic agents and wealth updates in machine learning markets
A.J. Storkey et al. (2012)

A machine learning market is a way of combining several estimators of the distribution of a random variable X : the possible outcomes, $1, 2, \dots, n$ correspond to Arrow-Debreu securities, the estimators are investors, each endowed with an initial wealth (confidence in the model) and trying to maximize their expected utility. The equilibrium can be estimated by tatonnement; for some utilities, it can be derived in closed form.

Image processing
G. Sapiro (Coursera, 2013)

1. The JPEG compression algorithm works as follows: convert the RGB image to YCbCr (the channels are more independent – this is just a linear transformation), cut the image into 8×8 blocks, apply a discrete cosine transform (DCT) to each of them, quantize the resulting coefficients (you can use more bits for the most important ones – the number of bits is specified by the “quantization matrix”; rescaling it changes the size and quality of the result), use Huffman encoding on the resulting stream of bits. (The DCT is actually the empirical Kahunen-Loève transform (KLT) from stochastic calculus, in the special case of Markovian images.)

Predictive encoding uses some algorithm to predict the $(n+1)$ st pixel from the previous ones and encodes the error. It can be used for lossless compression (JPEG-LS) or lossy video compression (MPEG)

2. Some image enhancement algorithms are straightforward to implement: curves (e.g., gamma correction), histogram equalization, histogram matching, local averaging (local mean, local median). *Non-local means* denoising finds and averages similar neighbourhoods (e.g., all the windows, all the gargoyles, etc.), in different parts of the picture.

Different noise removing filters are adapted to different types of noise: median filter for salt-and-pepper noise, the gaussian blur for gaussian noise, etc. To infer the type of noise, you can pick a small region of the image and assume that it was initially uniform; the noise may not be the same on the whole image.

Image degradation can be modeled as

$$g = f * h + \text{noise}$$

where f is the initial image, h the degradation and $*$ denotes convolution. If the noise is negligible, the Fourier transform gives $G = FH$, therefore $F = G/H$. To estimate h , calibrate the camera on a known, simple image f , e.g., a single dot. The *Wiener filter* takes the noise into account: $\hat{F} = H^*/(H^2 + S_{\text{noise}}/S_{\text{signal}})G$ where S is the power spectrum and $*$ the complex conjugate (it minimizes the expected square error). Since $S_{\text{noise}}/S_{\text{signal}}$ is usually not known, it is often replaced with a constant K (the gain): just multiply by $H^*/(H^2 + K)$.

3. The *Hough transform* detects lines (or other parametric shapes). Let X be the set of the pixels in the image and Y the set of the shapes we want to detect, e.g., the set of straight lines in X , discretized (and parametrized in some way, e.g., $\rho = x \cos \theta + y \sin \theta$). For each pixel in X , increase the intensity of the corresponding pixels in Y . The most prominent shapes are the pixels in Y with the highest intensity. To find segments, rather than (infinite) lines, check back on the image where there are pixels on the line

Otsu's image segmentation algorithm is simplistic: look at the histogram of the image and try to see two modes,

e.g., find the threshold that minimizes the within variance. Since the total variance is the within variance plus the between variance, you can instead maximize the between variance; as you move the threshold, it is easy to update. If the background is not uniform, it may not work: try to apply the algorithm on a moving window instead.

The *Mumford-Shah* image segmentation algorithm solves an optimization problem: find a new image, locally smooth, with a penalty for the difference with the initial image (MSE), and another penalty for the presence of a lot of edges. More precisely, find a new image J and the edge locations B to minimize

$$\alpha \int_D \|I - J\|^2 d\Omega + \beta \int_{D \setminus B} \|\nabla J\|^2 d\Omega + \gamma \int B d\Omega.$$

Image segmentation is often interactive: the application asks the user to indicate what is the foreground and what is the background, by a few strokes inside those regions. We can adapt Otsu's algorithm: look at the distribution of the colour of the pixels along those strokes, and assign each new pixel to the region whose distribution is closest. The *weighted distance transform* also uses geometric information: for each pixel, compute the weighted geodesic distance to the foreground (using Dijkstra's algorithm, in linear time) and the background scribble, and assign the pixel to the closest scribble; as weight, use the change in the foreground probability along the direction of the path, $|\nabla p_{\text{Foreground}} \cdot ds|$. You can refine the result by adding new scribbles, automatically, inside the detected foreground, and inside the detected background.

Segmentation can also be formulated as a graph-theoretic problem: consider the graph whose vertices are the pixels, plus one vertex for the background, and one for the foreground. The edges correspond to neighbouring pixels, with also one edge from the background to each pixel, and the same for the foreground. The graph is weighted: between pixels, use the absolute value of the difference between a pixel and its neighbours; between the foreground and a pixel, use the probability that a pixel belongs to the foreground, e.g., from the scribbles given by the user in interactive segmentation. This is a *min-cut* problem.

Those algorithms can be generalized to process video (e.g., Roto-brush video segmentation in Adobe After Effects): do not consider the image as a whole, but smaller windows covering it; take motion into account (the same window on the next frame is not exactly at the same place); use mixed models for the distribution of the colours of the background and foreground; use a shape prior and a colour prior to find the new segmentation, with more weight on the shape prior if the colours are not distinct enough.

4. Let C be a planar curve, s its arclength parametrization, C_s its first derivative (of length 1), C_{ss} its second derivative, orthogonal, of length κ (the curvature). Cartan's theorem states that, up to Euclidian transformation, a curve is entirely determined by its curva-

ture $(s, \kappa(s))$, seen as a function of the arc length. It can be generalized to equi-affine transformations, *i.e.*, affine transformations preserving areas (their determinant is 1): an equi-affine parametrization v is such that $\det(C_v, C_{vv}) = 1$; the equi-affine curvature μ is defined by $C_{vvv} = \mu C_v$; up to equi-affine transformation, a curve is entirely determined by its equi-affine curvature $(v, \mu(v))$, seen as a function of the equi-affine arclength v .

The evolution of a planar curve can be described by a partial differential equation (PDE),

$$\frac{\partial \text{Curve}}{\partial t} = V(p, t),$$

such as the curvature flow $C_t = \kappa n$ (or heat flow, $C_t = C_{ss}$, a simple curve becomes convex and turns into a circular point), the affine affine heat flow $C_t = \kappa^{1/3} n$ (elliptic points), the constant flow $C_t = n$. (The tangential components do not affect the geometry of an evolving curve – they just change the parametrization.) Geodesic *active contours* use $C_t = g\kappa n$, with $g = 1/\|\nabla I\|$. Curves can also be represented by a level set, $f(x, y) = 0$: the normal is $n = -\nabla f / \|\nabla f\|$, the curvature $\kappa = \text{div}(\nabla f / \|\nabla f\|)$ and the curve evolution $C_t = Vn$ becomes $f_t = V \|\nabla f\|$.

Calculus of variation refers to infinite-dimensional optimization problems: finding a function (e.g., a curve) that minimizes some quantity. For instance, if we want to find the function u (subject to some boundary conditions) to minimize $\phi(u) = \int F(x, u, u_x) dx$, the first order conditions are given by the *Euler-Lagrange equation*, $\phi'(u) = 0$, where ϕ' is the functional derivative, *i.e.*,

$$\frac{\partial F}{\partial u} - \frac{d}{dx} \frac{\partial F}{\partial u_x} = 0.$$

Solving the variational problem with the steepest descent method $u_{t+1} - u_t - \alpha \phi'(u)$ is tantamount to adding a time parameter t and solving the PDE $u_t = -\alpha \phi'(u)$. For instance, minimizing $\int \rho(\|\nabla I\|) d\Omega$ with $\rho(a) = a^2$ gives isotropic diffusion (gaussian smoothing, the heat equation) $I_t = \Delta I = \text{div}(\nabla I)$, and $\rho(a) = a$ gives anisotropic diffusion $I_t = \text{div}(\nabla I / \|\nabla I\|)$ – there is less diffusion when the gradient is high, *i.e.*, near the edges, which are preserved.

For instance, $\phi(u) = \int \|\nabla I\|^2 d\Omega$ gives $\phi'(u) = -2 \text{div} \nabla I = -2\Delta I$ – the corresponding diffusion (isotropic diffusion) is the heat equation, *i.e.*, Gaussian smoothing.

More generally, one can consider $\phi(I) = \int \rho(\|\nabla I\|) d\Omega$. In particular, $\phi(I) = \int \|\nabla I\| d\Omega = \iint \sqrt{I_x^2 + I_y^2} dx dy$ gives $\phi'(u) = -\text{div}(\nabla I / \|\nabla I\|)$: this anisotropic diffusion is similar to Gaussian diffusion, but with less diffusion near the edges (where the denominator $\|\nabla I\|$ is large).

The gradient ∇I gives some edge information, but only locally. To have global edges, start with some local

edge detection, e.g.,

$$g(x, y) = \frac{1}{1 + \|\nabla(G * I)\|^2}$$

($G * I$ is the smoothed image, G can be chosen to detect some specific shapes, and since the gradient is in the denominator, edge pixels have a low value) and find a curve C that minimizes $\int_C g(x, y) ds$; the resulting diffusion gives *active contours*. You need a starting curve to deform: take segments regularly placed on the image, or a closed curve around the boundary of the image.

Histogram equalization (contrast enhancement) can also be formulated as a diffusion:

$$\frac{dI(x, y)}{dt} = I(x, y) - \text{number of pixels of value} \geq I(x, y).$$

It minimizes

$$\frac{1}{2} \int (I - \frac{1}{2})^2 dx - \frac{1}{4} \iint (I(x) - I(y)) dx dy.$$

You can change the objective function, e.g., by giving more/less weight to some region.

Inpainting fills in the missing parts of an image (e.g., because they contained undesirable elements, or they were damaged or lost). This can be done with PDEs: assume that the laplacian ΔI does not change as we move along the edges, i.e., in a direction $\nabla^\perp I$ orthogonal to the gradient ∇I ; propagate according to

$$I_t = \nabla \Delta I \cdot \nabla^\perp I$$

(only the missing parts of I change).

Inpainting can also be done with calculus of variations: if we know the normalized gradient $\theta = \nabla I / \|\nabla I\|$, then $\theta \cdot \nabla I = \|\nabla I\|$. We can look for θ and I to minimize (on the region to fill in)

$$\int (\|\nabla I\| - \theta \cdot \nabla I) d\Omega;$$

the corresponding Euler-Lagrange equations are

$$\frac{dI}{dt} = \text{div} \frac{\nabla I}{\|\nabla I\|} - \text{div} \theta.$$

(In practice, one would add a regularizing term, e.g.,

$$\int (\text{div} \theta)^p (a + b \|\nabla(G * I)\|) d\Omega,$$

to the function to minimize.)

To preserve texture, while inpainting, decompose the image into a smooth (say, locally constant) component and a texture component, and inpaint both separately. To inpaint the texture, use smart cut-and-paste: find a similar region, elsewhere in the image (like non-local means), allowing for some deformation; do this for a covering by small regions, but require that the translation vector changes smoothly from one region to a

neighbouring one. This can be formulated in a variational way.

5. Noise can be removed from an image y by finding the image x that minimizes

$$f(x) = \|x - y\|^2 + \lambda G(x).$$

There are many popular choices for the penalty term (Bayesian prior) $G(x)$: $\|x\|^2$, $\|\Delta x\|^2$ (smoothing), $\rho(\Delta x)$ (robust statistics), $\|\nabla x\|$ (total variation), etc.

Sparse modeling tries to express an image as a linear combination of images (“atoms”) from a set of reference images (“dictionary” – it could be a basis, but it will usually be larger), with as few atoms as possible, i.e., with an L^0 penalty. Often, one can replace the L^0 pseudo-norm with the L^1 norm: the optimal solution is the same; the relaxed problem is convex (and a greedy algorithm, adding one element at a time, gives reasonably good results). The *k-SVD* algorithm learns the weights and the dictionary for a given set of images (usually, small overlapping patches in the same image) at the same time: start with a random dictionary (or, say, the DCT one used by JPEG), compute the weights, update the reference images, iterate. This can be used for denoising, inpainting, video inpainting, zooming, demosaicing (for camera sensors: for each pixel, only one of the three colours is present), etc.

Compressed sensing is another form of sparse modeling: it models each 8×8 patch as a mixture of (say) 10 Gaussians (these are 64-dimensional Gaussians: use low-rank variance matrices), alternates between the estimation of the Gaussians and the estimation of the weights of the patches (MAP-EM) and only uses one of the Gaussians for each patch.

Sparse modeling can also be used for classification – concatenate dictionaries adapted to the different types of scenes to recognize, and look at which atoms were used.

Artificial Intelligence Planning **G. Wickler and A. Tate (Coursera, 2013)**

1. A planning problem is described by a set of states, a set of actions, acting on the set of states, an initial state, and a goal state.

There are more parsimonious descriptions, with symbols (e.g., `robot1`, `box1`, `locationA`), predicates (e.g., `location(what, where)`), and parametrized actions (e.g., `move(who, what, from, to)`); each action has a set of (positive and negative) preconditions, and a set of (positive and negative effects). This can be extended with disjunctions and quantifiers in the preconditions. Those problems can be represented in a Lisp-like (PDDL) or Prolog-like (STRIPS) syntax.

Examples include puzzles (canibals and missionaries, 8-puzzle, 8-queen problem, etc.), autonomous systems (elevators, space probes, unmanned vehicles) and business process management.

There are many algorithms to solve those (NP-

complete) problems: state-space search (A* and variants, with a heuristic function), plan-space search, graphplan, reduction to other problems (SAT, ASP (answer set programming), integer programming).

2. State-space search algorithms rely on some graph search algorithm, often based on a heuristic (an approximation of the distance between a state and the goal), such as best-first-search (explore the node whose estimated distance to the goal is shortest) or A* (also include the distance from the initial state), in their tree-search or graph-search variants (if you assume that the graph is a tree, you do not have to check if a node has already been visited; if it is not, you should check, and the technical condition on the heuristic is slightly different).

There are memory-efficient variants of A*, such as iterative deepening A* (depth-first search with a bound on the cost, restart many times as you progressively increase the bound – since most of the work is done in the last iteration, it is not that slow).

Instead of *forward search* (going from the initial state to the goal), you can try *backward search* (going from the goal to the initial state), but in both cases, the branching of the tree can be too high. To reduce it, you can consider partially instantiated actions ($a(s, -)$, where “-” is left unspecified until needed).

But in the case of n independent actions that can be executed in any order, there are still $n!$ different paths...

3. *Plan space* search is another form of backward search, in which we progressively build the plan, by adding actions, but without imposing any order until absolutely necessary. More precisely, start with the goal, look at its preconditions, find one that is not satisfied, find an action that enforces it (you will need to backtrack on that choice) and add it as a causal link (“action a establishes precondition p of action b ”). Unsatisfied preconditions are only one of the two types of “flaws” a partial plan can have: there is a “threat” when the effect of an action contradicts a causal link – this can be remedied by imposing an order relation, putting the action before or after the causal link. (In addition, the actions added can be partially instantiated, e.g., `move(who, what=apple, from=tree, to=bag)`, and one progressively adds constraints ($=, \neq$) on unbound variables.) This is no longer a search in the state graph, but in the graph whose nodes are partial (and partially ordered) plans, and whose edges are partial plan refinements.

4. *GraphPlan* addresses the branching factor problem by considering an (easy-to-solve) relaxed problem and searching among the solutions of the relaxed problem.

The planning graph is a layered, directed graph:

- The first proposition layer P_0 contains the propositions valid in the initial state (we assume that there are no negative conditions);
- The action layer A_i contains all actions whose pre-

conditions form a subset of the previous proposition layer P_{i-1} ; also add no-op actions for each proposition in layer P_{i-1} ;

- The proposition layer P_i contains all the propositions in P_{i+1} plus all the positive effects (not the negative ones) of the actions on A_i (P_0 corresponds to a state, but later proposition layers no longer do: they usually contain incompatible propositions);
- Edges from the preconditions of an action (in the previous layer) to this action;
- Edges from an action to its positive and negative effects in the next layer (use two edge types).

This already gives a necessary condition for reachability. We can also add edges inside layers to indicate incompatibilities (*mutex*). Two propositions are non-mutex if they are produced by independent (non-mutex) actions (or by a single action) (you need to distinguish between positive and negative effects). Two actions are mutex if they are dependent, or if some of their preconditions are mutex. The layers are increasing in size, and the sets of mutexes are decreasing, so that the layers stop changing at some point.

5. To build a *heuristic* for the A* algorithm, you can try to simplify the problem in some way, by removing some of the information or constraints. For instance, for the 15 puzzle, you can ignore the labels on cells 8–15, and store the number of (marked) tiles to move in a *pattern database*. Even better, you can do this for cells 1-7 and 8-15 (disjoint pattern databases), and add the corresponding heuristics. (In general, to find disjoint pattern databases, try to find mutually exclusive sets of proposition symbols – e.g., 2 sets – if there are too many, the problems are trivial and not informative.

The traveling salesman problem (TSP) gives another example: when moving the truck, do not remove it from its old position (*ignore deletes*), so that it is in several positions at the same time (the relaxed problem reduces to the minimum spanning tree (MST) problem, which also gives a heuristic solution).

Heuristic search often wins the International Planning Competition.

The *FF planner*, and most state-of-the-art planners, use the planning graph on a relaxed problem (no deletes) to find a heuristic: it is not admissible, but it is usually good.

6. *Hierarchical task networks* (HTN) rely on more domain knowledge to group actions into higher level tasks: the problem is then to find a decomposition of a goal into tasks.

7. Planning problems can be augmented with resources, time (time points, or interval algebra), uncertainty (replace “state” with “sets of states”), probability (partially observable markov decision process (MDP)), interacting agents. Real-world planners need to monitor the plan execution and modify the plan if something unexpected happens.

When run on similar problems, the planner can learn

macro operations (the tasks in a HTN) and refine its heuristic function (to help choose between nodes with the same heuristic).

Private equity as an asset class
F. Fraser-Sampson (2010)

Private equity (PE) refers to equity (shares in a company) not listed on a stock exchange. A *GP* is a manager of a PE fund, an *LP* is an investor in a PE fund.

When investing in a PE fund, you pledge that you will invest a certain amount (the *committed capital*), in the coming years, when required (when the money is *called* or *drawn down* – you do not know the dates in advance). The *invested capital* will be lower – when you start to invest in PE, it will remain very low, for several years. The committed capital is usually 1.6 to 2 times the *allocated capital* (the proportion of your wealth you wish to eventually invest in PE).

Since the cash flows are random, and mostly negative, there is no good definition of annual returns. One can look at the *J-curve*, *i.e.*, the running internal rate of return (IRR), $IRR_{[0,t]} \sim t$, or various multiples:

$$\begin{aligned} \text{DPI} &= \frac{\text{Distributed}}{\text{Paid in}} \\ \text{DCC} &= \frac{\text{Distributed}}{\text{Committed capital}} \\ \text{PICC} &= \frac{\text{Paid in}}{\text{Committed}} \\ \text{RVPI} &= \frac{\text{Remaining value}}{\text{Paid in}} \\ \text{TVPI} &= \frac{\text{Distributed} + \text{Remaining}}{\text{Paid in}} \end{aligned}$$

Look at their distribution for a given sector, fund size, region, industry, vintage year (starting date of the fund), time since inception.

There is no useable notion of “risk” either.

[Apparently, no one tries to model the randomness of those cash flows – lack of data?]

A **buyout** fund takes a majority part in a company, using a lot of debt: since steady cash flows are needed to service the debt, the target is a mature company, with good cash flows and too little debt. Buyout returns are driven by earnings (strictly speaking, the cash flows, but the EBITDA is often used instead: it is available, does not include interest payments (they will change), taxes (they will change: debt is more tax-efficient), depreciation and amortization (mere accounting cookery, not real cash)), earnings growth, “multiple” (P/E, *i.e.*, price/earnings ratio, influenced by the market), leverage. In the valuation, do not forget to take inflation into account (or use a real currency).

When investing in a buyout fund, decompose the performance into the contribution of leverage, market and earnings growth as follows (with $\{i, j\} = \{1, 2\}$), and

compare with peers.

$$\begin{aligned} \text{Enterprise value} &= \text{Debt} + \text{Price} \\ &= \text{Debt} + \text{Earnings} \times \text{Multiple} \\ \Delta \text{EV} &= \Delta \text{Debt} \\ &\quad + \text{Earnings}_i \times \Delta \text{Multiple} \\ &\quad + \text{Multiple}_j \times \Delta \text{Earnings}. \end{aligned}$$

When choosing a buyout, check the accounting data and projections, the supply and distribution channels, the anti-trust legislation, the pension scheme (often in deficit).

Venture refers to a majority share in young companies, developing new applications of extant technologies (rather than new, unproven technologies: that would be too risky), usually in the IT or biotech industries. Most companies (95%) in which a venture fund invests fail – the performance of the fund is often due to a single company succeeding (its value increases 25-fold) – a *home-run*. Valuing those companies is problematic: data is scarce, accounting rules are murky (Europe) or just advisory (US). Each financing round changes the valuation and increases the dilution. When investing in a venture fund, look at its multiple and the proportion of home-runs; compare with the industry. When choosing a venture company, check the technical soundness of the project, the potential key customers, the key personnel, and intellectual property ownership.

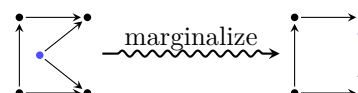
Development capital refers to a minority investment in an established company, with steady cash flows and a decent market share; the money is needed for a specific goal (e.g., to develop a new product) or because one of the current shareholders is leaving (retiring).

Growth capital is similar, but the goal is to grow the market share further, or maintain it in a growing market: since this requires extra marketing and product development, the profits will drop – but the sales and the company value will increase.

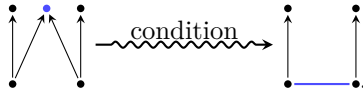
Both development capital and growth capital are short-term (3 years) minority investments: minority protection (e.g., negative control, *i.e.*, veto right) and exit protection are necessary.

Graphical Markov models
with mixed graphs in R
K. Sadeghi and G.M. Marchetti
(R Journal, 2012)

Directed acyclic graphs (DAG) describe conditional independence relations in an imperfect way. In particular, they are not stable under marginalization and conditioning. They can be generalized by adding *arcs* between response variables (when a common cause has been marginalized)



and *lines* between explanatory variables (when a common consequence has been conditioned on)



Ribbonless graphs are the smallest class of mixed graphs, containing DAGs, and stable by marginalization and conditioning. The notion of d -separation (to read independence relations off the graph) can be generalized to m -separation.

The **ggm** package can manipulate those graphs (**gRain** dealt with DAGs and d -separation, **graph**, **igraph**, **gRbase** are more general-purpose).

Rfit: rank-based estimation for linear models
J.D. Kloeke and J.W. McKean
(R Journal, 2012)

Rank regression estimators are

$$\hat{\beta}_\phi = \underset{\beta}{\operatorname{Argmin}} \|y - X\beta\|_\phi$$

where

$$\|\varepsilon\|_\phi = \sum_{i=1}^n \phi \left(\frac{\operatorname{Rank} \varepsilon_i}{n+1} \right).$$

The crs package: nonparametric regression splines for continuous and categorical predictors
Z. Nie and J.S. Racine (R Journal, 2012)

There are many types of splines: *smoothing splines* (penalized splines) use observations as potential knots; *regression splines* use equidistant or equiquantile knots. Discrete predictors can be accommodated with **kernel weighting**: replace the indicator functions $\mathbf{1}_{X=x}$ (which take two values, 0 and 1) with $\lambda^{\mathbf{1}_{X=x}}$ (which take two values, 1 and λ) – for ordered categorical variables, use $\lambda^{|X-x|}$ instead.

The splines come from GSL (GNU scientific library) and the metaparameters are estimated by cross-validation with the NOMAD black box optimizer (mesh adaptive direct search, MADS).

influence.ME: tools for detecting influential data in mixed effects models
R. Nieuwenhuis et al. (R Journal, 2012)

This package computes the influence (DF betas, Cook's distance, percentile change $(\hat{\beta} - \hat{\beta}_{-j})/\hat{\beta}$, change in significance) of observations, or groups of observations, in mixed effects models (**lme4**).

Strategic risk management and risk monitoring for pension funds
S. van Hoogdaem et al.

Asset liability management (ALM) should use the same tools as traditional (1-period) portfolio management:

performance attribution, risk decomposition, stress tests, etc.

Balanced baskets: a new approach to trading and hedging risks
D.H. Bailey and M.L. de Prado (2012)

The many variants of the minimum variance portfolio can be used to hedge a position (by fixing one of the weights, say, w_1).

- Minimum variance: $\operatorname{Argmin}_w w'Vw$, i.e.,

$$\operatorname{Argmin}_w \operatorname{Var} \left[\sum_i w_i X_i \right]$$

- Maximum diversity:

$$\operatorname{Argmax}_w \frac{\sum_i w_i \operatorname{Sd} X_i}{\operatorname{Sd} \sum_i w_i X_i}$$

- Equal risk:

$$\forall i, j \quad w_i \frac{\partial \operatorname{Risk}}{\partial w_i} = w_j \frac{\partial \operatorname{Risk}}{\partial w_j},$$

where $\operatorname{Risk} = \sum_i w_i \frac{\partial \operatorname{Risk}}{\partial w_i}$ by Euler's formula.

There is a variant of the equal risk portfolio, the *diversified risk portfolio*, which asks for an equal contribution of each principal component (or each risk factor, if you have a risk model) – since the largest components correspond to the main sources of risk, you probably do not want them in a hedging basket – but it makes sense for a trading basket.

- Equal correlation:

$$\forall j, k \quad \operatorname{Cor} \left(\sum_i w_i X_i, X_j \right) = \operatorname{Cor} \left(\sum_i w_i X_i, X_k \right)$$

- Minimax subset correlation:

$$\operatorname{Argmin}_w \max_{\substack{I \subseteq [1, n] \\ I \neq \emptyset}} \left| \operatorname{Cor} \left(\sum_j w_j X_j, \sum_{i \in I} w_i X_i \right) \right|$$

Some of those approaches can also be used to build trading (rather than hedging) portfolios, e.g., the maximin subset correlation portfolio,

$$\operatorname{Argmax}_w \min_{\substack{I \subseteq [1, n] \\ I \neq \emptyset}} \left| \operatorname{Cor} \left(\sum_j w_j X_j, \sum_{i \in I} w_i X_i \right) \right|.$$

The article also provides Python implementations.

Beware of the DAG!
A.P. Dawid (2009)

One could be tempted to interpret directed acyclic graphs (DAGs) representing graphical models as describing causal relations: they only describe a set of conditional independence relations (or, alternatively, a factorization of the joint probability distribution), and the direction of the edges is not that meaningful.

Causal independence $X \perp\!\!\!\perp Y$ implies conditional independence $X \perp\!\!\!\perp Y \mid Z$, but the converse does not hold: some causes could cancel out, or there may be confounding (overlooked) causes.

To have a causal interpretation of a DAG, one can add two types of nodes: intervention nodes (whose value can be chosen – passively observing the system is not sufficient for causal discovery: this is the difference between observational and experimental sciences) and confounding nodes (for each set S of nodes on the original graph, add a node with edges towards each node in S – that is a lot of nodes, but for a given analysis, only a handful is usually needed).

The best you can do with a DAG with no observational nodes is apply Occam’s razor: the conditional independence relations are consistent with many causal interpretations; choose the most parsimonious, e.g., to choose between

$$p(x, y) = p(x)p(y|x)$$

and $p(x, y) = p(y)p(x|y),$

pick the relation with the lowest algorithmic complexity.

The big data bootstrap
A. Kleiner et al.

The *bootstrap* (resampling with replacement) is expensive for large datasets (you have to build new datasets of size n): to *subsampling* (resample $m \ll n$ observations without replacement) or *m out of n bootstrap* (resample $m \ll n$ observations with replacement), prefer the *bag of little bootstraps* (take s subsamples of size $m \ll n$, e.g., $m = n^{0.6}$, resample them r times with replacement to form datasets of size n , not m – since there are many duplicates, you just need to keep track of the number of occurrences of each observation): it is more robust and has better theoretical properties.

A scalable bootstrap for massive data
A. Kleiner et al. (2012)

More details, on the choice of the hyperparameters (e.g., stop using a subsample when the corresponding estimate ceases to change), with applications to time series (e.g., the stationary bootstrap: take a block of length $m \ll n$, take a point at random, with probability $1 - p$ take the next point, with probability p take another point at random, iterate with more points, repeat with more blocks).

Improving GARCH volatility forecasts with regime-switching GARCH
F. Klaassen (2001)

GARCH forecasts are too high in volatile periods; they can be improved by using a regime-switching GARCH model (with two regimes). The straightforward model,

$$\sigma_{1,t}^2 = \omega_1 + \alpha_1 \varepsilon_{t-1}^2 + \beta_1 \sigma_{1,t-1}^2$$

$$\sigma_{2,t}^2 = \omega_2 + \alpha_2 \varepsilon_{t-1}^2 + \beta_2 \sigma_{2,t-1}^2,$$

is difficult to estimate. Instead, one can replace $\sigma_{i,t-1}^2$ in the right handside with

$$E_{t-1}[\sigma_{i,t-1}^2] = p\sigma_{i,t+1}^2 + (1 - p)\sigma_{2,t-1}^2$$

or, better, $E[\sigma_t^2 | r_t = i]$.

A review of backtesting and backtesting procedures
S.D. Campbell (2005)

Regulators check the validity of a VaR (value-at-risk) model by counting the number of exceedances (violations) in the past year (and they increase the market risk capital accordingly). The violations should be Bernoulli, independent: one can check if they have the correct distribution (`prop.test`, `binom.test`), or if they are independent (independence χ^2 on the contingency table of consecutive observations, runs test – but this only tests for very short term independence), or both (χ^2 test on the same table).

One could also convert the observations to p -values for the VaR model and check if they are uniform $U(0, 1)$ and independent. Since we are only interested in the tail, we could bin those values (say, $[0, 1\%]$, $[1\%, 2\%]$, $[2\%, 5\%]$, $[5\%, 100\%]$) and perform a χ^2 test (we expect $(\beta - \alpha)N$ observations in $[\text{VaR}_\alpha, \text{VaR}_\beta]$).

Those tests may have low power and may fail to identify underreporting.

One could also use a loss function, e.g., that defining the expected shortfall – but it will just tell you that VaR estimators are bad CVaR estimators. (But it makes sense for the regulators to do that: this is how they use the VaR – to compute the risk capital.)

Extended analysis of backtesting frameworks for value at risk
G.J. van Roekel (2008)

To test your value at risk (VaR) model, use the following tests, after checking their power, *i.e.*, the proportion of type I and type II errors, for different null ($p = 1\%$, $p < 1\%$) and alternative ($p \neq 1\%$, $p > 1\%$) hypotheses.

- Test the number of exceedances, with an exact test or a likelihood ratio (LR) test, test the time to the next one.
- Test their independence, with an LR test, using indicator functions “exceedance in $[t - k, t + k]$ ” rather than “exceedance at t ”; test the distribution of the

time between two exceedances: it should be geometric, *i.e.*, easy to approximate with an exponential distribution (for instance, you could test the null that the distribution is exponential vs the alternative that it is Weibull but not exponential). Tests for discrete distributions have more power.

- Compare the regulatory capital (derived from the VaR) with the average expected shortfall (CVaR).

A guide to modelling counterparty credit risk **M. Pykhtin and S. Zhu (2007)**

The *counterparty exposure* of an asset of value v is v_T (its value when the counterparty defaults, at some time T in the future). Using simplified pricing models, you can simulate possible evolutions of v_t and compute its expectation and quantiles (for instance, for bonds or swaps, it progressively increases, because of uncertainty, with a drop at each cash flow, because the remaining cash flows are decreasing). The *credit value adjustment* (CVA) is the risk-neutral expectation of the loss.

Algorithms, design and analysis II **T. Roughgarden (Coursera, 2012)**

The second part of the course covered greedy algorithms, dynamic programming and NP-completeness.

Greedy algorithms never come back on a decision they made, and therefore run in linear time: e.g., optimal caching (remove from the cache elements you will not need any time soon), scheduling (each job has a weight and a duration, sort them by weight/duration). If they happen to be correct (rarely), this can often be proved by induction or with an *exchange argument* (start with an optimal solution and transform it into the output of the algorithm, without ever making it worse).

For the minimum spanning tree (MST) problem, Prim's algorithm grows a connected tree (and can be efficiently implemented with a heap), while Kruskal's algorithm grows a forest by adding the cheapest edges as long as they do not introduce a cycle (and can be implemented with a *union-find* data structure). They are correct and fast, but there are faster algorithms. Kruskal's MST algorithm can be used to find the k -clustering of a set of points that maximizes spacing (the spacing is $\min d(a, b)$ where a and b are in different clusters) *i.e.*, the distance between the most alarmingly close points that are not in the same cluster.

Huffman codes (binary prefix-free codes; they can be represented as binary trees) are also built greedily, bottom-up: start with each symbol in a 1-element tree, merge the two least frequent elements, iterate.

Dynamic programming solves a problem by considering sub-problems, with an order relation (often inclusion), from whose solutions it is possible to solve the original problem. Contrary to divide-and-conquer, the sub-problems are overlapping, and it is necessary to cache (memoize) the intermediate results. Here are a few examples.

- Independent set (a set of vertices, no two of which are adjacent) of maximum weight in a path-graph (a graph that is also a path, *i.e.*, a tree of arity 1) [hint: either the last element is in the set or not];
- Knapsack with integral sizes [same hint];
- Sequence alignment;
- Optimal binary search tree (minimum average search time, when the frequency of each term is known);
- For the single-source shortest path problem, use Dijkstra's algorithm, if there are no negative edges, or Bellman-Ford's algorithm (slower): look at shortest paths from s with at most k edges (you can save on space by keeping a pointer to the previous vertex in the shortest path) – a distributed version is used for internet routing;
- For the all-pairs shortest paths problem (APSP), the Floyd-Warshall algorithm computes the length of the shortest path from i to j passing through (at most) $1, 2, \dots, k$; Johnson's algorithm modifies the weights, by adding them $p_i - p_j$, where p_i is the length of the minimum path from a new vertex to i : this makes them nonnegative, so we can use Dijkstra's algorithm.

A problem A reduces to a problem B , denoted $A \preceq B$, if a polynomial algorithm for B gives a polynomial algorithm for A . If \mathcal{C} is a class of problems, a problem $A \in \mathcal{C}$ is \mathcal{C} -complete if all problems in \mathcal{C} reduce to A , *i.e.*, A is the most intractable problem in \mathcal{C} (but there can be many problems as difficult as A). Let P denote the class of problems solvable in polynomial time. Let NP be the class of problems for which it is possible to verify a solution in polynomial time, and whose solutions have polynomial lengths; intuitively, these are the problems solvable by brute-force search. It is conjectured, but not proved, that $P \neq NP$.

There are many NP-complete problems: TSP, 3SAT, etc.

To solve an NP-complete problem, you can:

- Focus on solvable special cases, e.g., the 2-SAT problem: when solving a 3-SAT problem, you may be able to isolate 10 or 20 variables so that, when you fix them, the problems becomes a 2-SAT problem: you can then combine brute force search on those variables and a polynomial 2-SAT algorithm;
- Use heuristics: for the knapsack problem, there are greedy (sort the items by decreasing value/weight – to prove that the the solution is at least 50% of the value of the optimal solution, consider a “fractional solution”, in which you can cut the objects) and dynamic-programming-based (rounding) heuristics;
- Find an exact algorithm, exponential-time but faster than brute-force search: for instance, dynamic programming for the knapsack problem, dynamic programming for the traveling salesman problem (2^n instead of $n!$: consider $L(S, j)$, the minimum length of a path from 1 to j that passes exactly once through each vertex in S , for all S containing 1 and j), etc.

In the vertex cover problem, you are asked to find a minimum set of vertices so that each edge has at least

one extremity in it. It can be solved in special cases, e.g., trees (dynamic programming) or bipartite graphs (maximum flow problem); for small graphs, use the fact that G has a vertex cover of size k iff G_u or G_v has a vertex cover of size $k-1$, where G_u is G without vertex u , and $u-v$ is an edge.

The maximum cut problem (finding the cut with the largest number of edges) is NP-complete. It is tractable in special cases: for instance, in bipartite graphs, there is a cut with all edges, and it can be found by breadth-first search in linear time. In general, you can use a *local search* heuristic: let $c(v)$ (resp. $d(v)$) be the number of edges from vertex v that cross (resp. do not cross) the cut, start with an arbitrary cut, move v if $d > c$. Use multiple random restarts to help improve the solution.

The 2-SAT problem (find an assignment of boolean variables so that some statement of the form $\bigwedge_k \varepsilon_k a_{i_k} \vee \eta_k a_{j_k}$, where ε_k, η_k are the identity or the negation) can be solved in polynomial time (only the 3-SAT problem is NP-complete):

- By considering the graph whose vertices are the variables and their negations, with an edge for each implication implied by the clauses (each is equivalent to two implications), the problem reduces to finding the strongly connected components;
- Backtracking (each clause forbids one assignment of the variables);
- Randomized local search (find an arbitrary unsatisfied clause and flip one of the variables, at most $2n^2$ times, and use $\log_2 n$ random restarts: the probability of finding an assignment if there is one is at least $1 - 1/n$ – this estimate is very conservative, but the algorithm is quadratic).

The following topics were not covered: maximum flow, linear programming, computational geometry, parallel or distributed algorithms, algorithms that run forever, streaming algorithms (when the data is too large to be kept in memory), etc.

Heterogeneous computing W.-M. W. Hwu (Coursera, 2012)

GPU programming is getting more popular, and the main API is currently CUDA, targeted at NVidia chips.

A CUDA program is just a C program, with a few more keywords. The memory can be allocated either on the host (CPU, `malloc`, `free`) or the device (GPU, `cudaMalloc`, `cudaFree`), can be copied between them (`cudaMemcpy`). Functions can be declared as callable and/or runnable from/on the host and/or the device: kernel functions, prefixed with `__global__`, are called from the host as

```
vecAddKernel<<<blocks,threads>>>(x,y,z, n);
```

and run on the device, in parallel (`blocks` \times `threads` threads); they do not return any value, but can change

the data in the arrays pointed to by their pointer arguments; they can retrieve the thread block and thread number through implicitly-defined variables `threadIdx.x`, `blockDim.x`, `blockIdx.x`.

Threads and blocks are arranged in 1-, 2- or 3-dimensional arrays. Threads in the same block can share memory: declare it as `__shared__` in the kernel function. Threads in a block can wait for other threads in the same block to reach the same point, by calling `__syncthreads()`. Shared memory and thread blocks can reduce the amount of data transferred between the host and the device, and greatly increase the program speed – the typical example is *tiling matrix multiplication*, i.e., multiplying block matrices. Convolution (or solving PDEs, e.g., wave propagation) requires *halo* data: data from neighbouring nodes, close to the boundary – you may want to process it in priority to avoid delaying the neighbouring nodes.

Many linear algorithms, such as the sum or the cumulated sum of a vector, can be executed in log-linear time (*parallel prefix sum*, or scan), by arranging the computations in a tree (dividing the number of threads by 2 at each iteration).

There are other “parallel design patterns” such as compaction (the use of sparse matrices), binning (the Barnes-Hut algorithm, for the N -body problem), scatter-to-gather conversion (map/reduce), etc.

Streams allow you to transfer data and compute at the same time. Manually managing them can be tricky.

MPI is a framework for distributed programming, where processes (possibly running on different machines) send and receive messages. The nodes can use CUDA.

OpenCL is similar to CUDA, but not limited to a single vendor, and a bit clunkier (the device code is compiled at runtime and has to be put in a string (sic); it cannot be called as a normal function: its arguments have to be pushed on a stack one by one; etc.).

OpenACC just adds a few `#pragmas` to a sequential C program:

```
#pragma acc parallel loop copyin(M[0:m*n])
                    copyin(N[0:n*p]) copyout(P[0:m*p])
for(int i=0; i<m; i++){
    #pragma acc loop
    for(int j=0; j<p; j++){ ... }
}
```

It is possible to fine-tune the number of workers (threads) and gangs (thread blocks).

C++Amp is a Microsoft-specific extension to C++ providing a `parallel_for_each` loop, synchronization, and implicit data transfer between host and device.

OpenMP and Thrust were not covered.

Software testing (Udacity, 2012)

Review of:

- The various types of tests: assertions, black box, white box, unit, integration, validation, differential, stress, random;
- Coverage metrics: lines of code; statements; loop (each loop is executed 0 times, once, several times); paths; modified condition/decision coverage (MC/DC): each condition in a decision is shown to independently affect the outcome of the decision; boundary value; synchronization (each lock does something); interleaving;
- Oracles (*i.e.*, ways of checking the output): weak oracles check for crashes, exceptions, failures in a controlled environment (valgrind), assertions; a strong oracle compares the output with another implementation, or using an inverse function to recover the input, or changes the (random) input in a way that should not change the output (null-space transformation).

The emphasis was on random testing:

- Automated whitebox coverage testing: start with some input, check why the coverage is not 100% (modify the input to increase coverage (it is a constraint satisfaction problem), iterate);
- Fuzzing: throw random data (GUI events, network activity, file contents) and see if the application crashes, often to find exploitable security faults; to fuzz implicit inputs (e.g., timing), just overload the machine; for a multi-threaded program, you can also add a `sleep()` before/after each synchronized operation;
- Random testing: like fuzzing, after fine-tuning the probability of the random data to increase code coverage and make the input more realistic – otherwise, the program notices that the input is invalid and exits: most of the code is not run; the random input generation can often be described with a finite automaton.

Software debugging (Udacity, 2012)

The word “bug” is ambiguous: there is a difference between *defect* (the code does not do what it should), *infection* (the program state is wrong) and *failure* (this becomes visible to the user).

Debugging should be done explicitly, in a scientific way: if it takes more than 5 minutes, write down what you are doing, what you think the code is doing, test that it is indeed the case. This allows you to resume debugging should you be interrupted, and prevents you from running in circles.

It is very easy to write debugging tools in Python, with `sys.settrace`: running the code step by step, following the changes of a variable, etc.

Delta debugging is an automated way of simplifying a

failing test: split the input into n subsets, try to remove one or more of them, and check if the test still fails; repeat with other values of n if needed. You can use tokens instead of characters, to make the delta debugger more aware of the structure of the data. `git-bisect` does something similar on the code, rather than the input.

You should check the invariants of your data structure at the beginning of each public method. It is possible to automatically discover invariants: start with variable ranges, variables types, variable relations ($=$, \leq , between 2 variables – or more), etc. (cf. Daikon).

There are various types of bugs: Bohr bugs (repeatable), Heisenbugs (they disappear or change when you try to reproduce/probe/isolate them, e.g., the program fails in the field, but works in the debugger; they can be caused by malware, the many unspecified behaviours in the C/C++ standards, etc.), Mandelbugs (chaotic, very complex causes), Schrödingbugs (they do not manifest themselves until someone reads the code and realizes it should never have worked).

To find the cause of bug, you can reason from the failure to its cause, or from the inputs to the failure. (Part of this process can be automated: the computer can find the conditions under which some assertion fails.) A *backward slice* is everything a statement depends on (recursively), a *forward slice* contains all statements that depend on a specific statement; a *dynamic slice* is a slice of an execution (a trace), rather than of a program. Slices can be computed automatically and give a small subset of the program, with only the lines executed and influencing the output.

Reproducing failures, when the inputs are complicated, can be tricky: files, GUI interaction (have the widget record the events they receive), OS (record the output of the API calls and replay them), randomness (set the seed), scheduling (find a way to make thread scheduling deterministic), physics (e.g., the OS or processor can behave differently when running on battery). Look for correlations (the phi coefficient) between lines executed, functions called, return value (zero/non-zero, sign, etc.), etc. and the failure.

If the version control system and the bug database are linked, you can mine the data they contain, to identify the most bug-prone parts of the code.

Quantum computation U. Vazirani (Coursera, 2012)

Quantum computation is just linear algebra on a finite-dimensional complex vector space, with slightly unusual terminology and notation.

A quantum bit (*qubit*, quantum state) is an element of \mathbf{C}^2 of norm 1. The standard basis is noted $|0\rangle$, $|1\rangle$, instead of the more usual (e_1, e_2) or $((1, 0), (0, 1))$: these are the *classical states*. Similarly, a vector $u \in \mathbf{C}^2$ (more precisely, a state, *i.e.*, a unit vector, $u \in \mathbf{S}_{\mathbf{C}^2}$) will often be written $|u\rangle$ rather than \mathbf{u} or \vec{u} . The dual (complex conjugate) of $|u\rangle$ is written $\langle u|$. The projec-

tion on $\text{Span}|\psi\rangle$ is $p_\psi = |\psi\rangle\langle\psi|$. The scalar product of $|u\rangle$ and $|v\rangle$ is written $\langle u|v\rangle$. *Quantum superposition* refers to the fact that $|0\rangle$ and $|1\rangle$ are not the only possible states: any linear combination (of norm 1) is possible.

A *measurement* (in \mathbf{C}^2) is another orthonormal basis, $(|u\rangle, |u^\perp\rangle)$; after a measurement, the system that was in state $\psi \in \mathbf{C}^2$ is now in state u with probability $|\langle u|\psi\rangle|^2$ and in state u^\perp with probability $|\langle u^\perp|\psi\rangle|^2$. In particular, after the measurement, you can no longer recover any information about the previous state $|\psi\rangle$: a lot of information has been lost.

The *Hadamard basis* (or sign basis) is

$$\begin{aligned}|+\rangle &= \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle \\ |-\rangle &= \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle.\end{aligned}$$

If you measure $|+\rangle$ or $|-\rangle$ in the $(|0\rangle, |1\rangle)$ basis, you cannot distinguish the two states: the measurement is $|0\rangle$ or $|1\rangle$, each with probability $\frac{1}{2}$, in both cases.

The *uncertainty principle* says that you cannot know both the bit ($|0\rangle$ vs $|1\rangle$) and the sign ($|+\rangle$ vs $|-\rangle$) of a qubit with certainty.

The *evolution* of a quantum system is described by a unitary operator, such as

$$\begin{aligned}H &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} && \text{Hadamard gate} \\ U_\theta &= \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} && \text{Rotation gate} \\ \text{NOT} &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} && \text{NOT gate} \\ Z &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} && \text{Phase flip gate.}\end{aligned}$$

The NOT gate flips $|0\rangle$ and $|1\rangle$; the phase flip gate flips $|+\rangle$ and $|-\rangle$; the Hadamard gate flips the canonical and sign bases; one has $Z = HXH$.

Qubits do exist in the real world: polarization of a photon ($\cos(\theta)|0\rangle + \sin(\theta)|1\rangle$) corresponds to a polarization of angle θ ; spin; particle with two states (ground $|0\rangle$ and excited $|1\rangle$), etc.

Consider a photon, polarized as $|+\rangle$, passing through a $(|0\rangle, |1\rangle)$ polarizing filter: it is measured, giving $|0\rangle$ or $|1\rangle$, each with probability $1/2$, and discarded if $|0\rangle$ – it exits as $|1\rangle$ with probability $1/2$. Then, let it go through another filter, tilted at 45 degrees, *i.e.*, $(|+\rangle, |-\rangle)$: the probability that it exits is also $1/2$, and it will be in state $|+\rangle$. If you stack n such filters, the probability that the photon remains at the end is $1/2^n$ – classical physics suggests 0 if $n \geq 2$.

The states of a system of two quantum bits are unit vectors in the tensor product $\mathbf{C}^2 \otimes \mathbf{C}^2$. The standard basis is $(|00\rangle, |01\rangle, |10\rangle, |11\rangle)$. Elements of the form $|u\rangle \otimes |v\rangle$ are called *decomposable*, the others are *entangled* – giving the state of a system made of two qubits is

not the same as giving the state of two 1-qubit systems: $\mathbf{S}_{\mathbf{C}^2 \otimes \mathbf{C}^2} \neq \mathbf{S}_{\mathbf{C}^2} \times \mathbf{S}_{\mathbf{C}^2}$; in general, $\mathbf{S}_{(\mathbf{C}^2)^{\otimes n}} \neq (\mathbf{S}_{\mathbf{C}^2})^n$. The *Bell state* is one such state (the two particles are in the same state):

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle = \frac{1}{\sqrt{2}}|uu\rangle + \frac{1}{\sqrt{2}}|u^\perp u^\perp\rangle.$$

Quantum computation is often said to be “exponentially powerful”: this power comes from the tensor product, $\dim(\mathbf{C}^2)^{\otimes n} = 2^n$, where classical systems have $\dim(\mathbf{C}^2)^n = 2n$. But once a measurement is made, the entanglement (and the exponential power) is destroyed.

The no-cloning theorem states that you cannot copy a qubit, *i.e.*, there is no unitary transformation U such that $U(|\psi\rangle \otimes |0\rangle) = |\psi\rangle \otimes |\psi\rangle$ for all states $|\psi\rangle$. But quantum teleportation is possible:

$$\exists U \quad \forall \psi \quad \exists u \quad U(|\psi\rangle \otimes |00\rangle) = |u\rangle \otimes |\psi\rangle;$$

to build this transformation, use the CNOT gate, which sends the canonical basis to a basis of Bell states.

We have only defined measurements in \mathbf{C}^2 , *i.e.*, for single qubits. In general, a *measurement* (or *observable*) is a hermitian operator A . From the spectral theorem, it can be written as

$$A = \sum_i \lambda_i |\phi_i\rangle\langle\phi_i|,$$

where the ϕ_i are a basis of eigenvectors with eigenvalues λ_i . The measurement of a quantum state $\psi = \sum_i \alpha_i |\phi_i\rangle$ is λ_i with probability $|\alpha_i|^2$.

The expectation and standard deviation of the measurement X of observable M on state $|\psi\rangle$ are

$$\begin{aligned}\mu &= \langle\psi|M|\psi\rangle \\ \sigma^2 &= E[X^2] - E[X]^2 \\ &= \langle\psi|M^2|\psi\rangle - \langle\psi|M|\psi\rangle^2.\end{aligned}$$

Those quantities can be interpreted as follows. Consider the distribution as a thick piece of wood (the area under the curve, thickened); the expected value is the point at which you can balance this piece of wood, as a see-saw; when you try to rotate this piece of wood (around a vertical axis, positionned on the mean), it offers some resistance (momentum of inertia), the same as that from two point masses, at $\mu + \sigma$ and $\mu - \sigma$. They also appear in the uncertainty principle:

$$\Delta A \Delta B \geq \frac{1}{2} |\langle\psi|[A, B]|\psi\rangle|$$

where

$$\begin{aligned}\Delta A &= \sqrt{\langle\psi|A^2|\psi\rangle - \langle\psi|A|\psi\rangle^2} \\ [A, B] &= AB - BA.\end{aligned}$$

The evolution of a quantum system is governed by Schrödinger's equation

$$i\hbar \frac{d|\psi(t)\rangle}{dt} = H|\psi(t)\rangle,$$

where H is an observable (a hermitian operator, also called “energy” – its eigenvalues are the energy levels); solving the equation gives

$$|\psi(t)\rangle = e^{-itH/\hbar} |\psi(0)\rangle$$

where $U = e^{-itH/\hbar}$ is a unitary operator (and \hbar is only here to keep the units consistent).

Here are a few building blocks to design quantum algorithms.

The Hadamard transform creates superpositions,

$$|0 \cdots 0\rangle \mapsto \frac{1}{2^{n/2}} \sum_{x \in \{0,1\}^n} |x\rangle,$$

on $(\mathbf{C}^2)^{\otimes n}$, and can be used to build other superpositions. For instance,

$$|\phi\rangle = \frac{1}{\sqrt{k}} \sum_{x \in f^{-1}(\{a\})} |x\rangle,$$

for $f : \{0,1\}^n \rightarrow \{0,1\}^n$ can be constructed as follows: start with $|0 \cdots 0\rangle |0 \cdots 0\rangle$, apply the Hadamard transform on the first half, to get $\frac{1}{2^{n/2}} \sum_x |x\rangle |0 \cdots 0\rangle$, apply f , to get $\frac{1}{2^{n/2}} \sum_x |x\rangle |f(x)\rangle$, and measure the second half; if the result is a (you do not get to choose a : it is random, in the image of f), the state becomes (proportional to)

$$\sum_{x \in f^{-1}(\{a\})} |x\rangle |f(x)\rangle.$$

The Hadamard transform is a special case of the *quantum Fourier transform* (QFT),

$$\sum_{k=0}^{n-1} \alpha_k |k\rangle \mapsto \sum_{\ell=0}^{n-1} \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \alpha_k e^{i \frac{2\pi}{n} k \ell} |\ell\rangle$$

on \mathbf{C}^n , with basis $|0\rangle, |1\rangle, \dots, |n-1\rangle$. If you view $\sum_k \alpha_k |k\rangle$ as a function $k \mapsto \alpha_k$, it is the classical discrete Fourier transform.

To find the period r in a “periodic state”

$$\sqrt{\frac{r}{n}} \sum_{k=0}^{n/r-1} |kr + \ell\rangle,$$

take the QFT (the offset ℓ disappears, the phase changes, and the period becomes n/r), measure (you get a random multiple of n/r), repeat, and take the gcd of those measurements: with enough of them, you can recover n/r . If $r \nmid n$, it is trickier, but it is still possible to recover enough information.

Those ideas can be applied to factor a number n (Shor’s algorithm): pick $x \in \mathbf{Z}/n$ at random, consider

$$f : \begin{cases} \mathbf{Z}/m \longrightarrow \mathbf{Z}/n \\ a \longmapsto x^a, \end{cases}$$

with $m \gg n$, build the superposition

$$\sum_{a \in f^{-1}(\{y\})} |a\rangle = \sum_{a : x^a = y} |a\rangle,$$

it is periodic; find the period r ; use it to factor n , by letting $y = x^{r/2}$, so that $y^2 = 1$ (in \mathbf{Z}/n), i.e., $n|(y-1)(y+1)$ (if r is odd or $y = -1$, start again).

For another, readable presentation of those topics, check S. Aaronson’s *Quantum computing since Democritus*.

Visualizing data using *t-SNE*

L. van der Maaten and G. Hinton (2008)

Stochastic neighbour embedding (SNE) is a dimension reduction technique trying to make the low-dimension conditional probabilities

$$q(j|i) \propto e^{-\|y_i - y_j\|^2}$$

as close as possible from the high-dimensional ones

$$p(j|i) \propto e^{-\|x_i - x_j\|^2 / 2\sigma_i^2},$$

where the variance σ_i^2 at point i is chosen so that the entropy at each point

$$H_i = - \sum_j p_{j|i} \log_2 p_{j|i}$$

be the same (and equal to some user-specified value – the *perplexity*, or effective number of neighbours, is 2^{H_i}), for the Kullback-Leibler distance

$$\sum_{i \neq j} p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}.$$

The Kullback-Leibler distance is asymmetric: there is a large cost for close points ending up distant, but a small cost for distant points ending up close. The gradient of the objective function is easy to compute, and the optimization can use momentum (to escape local extrema) and noise (of decreasing variance, as with simulated annealing).

The *t-distributed SNE* (*t-SNE*) algorithm uses the joint probabilities q_{ij} instead of the conditional ones $q_{j|i}$ (for the inputs, if i or j is an outlier, $p_{ij} \approx 0$, so one may prefer to cheat and use the symmetrized conditional probabilities

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2}$$

instead), and replaces the Gaussian distribution e^{-d^2} with a Student (Cauchy) one $(1+d^2)^{-1}$. The optimization uses gradient descent, with momentum (small at the beginning), an L^2 penalty $\sum_i \|y_i\|^2$ and early exaggeration (multiply the p_{ij} by 4, and do not worry if they do not sum up to 1) to help separate the clusters.

If there are clusters in the data, they are much more clearly separated than with other methods such as *Isomap* (find the nearest neighbours of each point, build the corresponding neighbourhood graph, use the geodesic distance on the graph), *local linear embedding* (LLE) (express each node as a weighted average of its nearest neighbours, find points in a lower-dimensional

space with similar weights – it reduces to a linear algebra problem with sparse matrices), *maximum variance unfolding* (MVU) (build the neighbourhood graph, preserve the distances between neighbours and maximize the distances between non-neighbours – this reduces to a semi-definite optimization problem).

For large datasets, build the neighbourhood graph, pick landmarks (*i.e.*, a small number of points), and estimate the conditional probabilities $p_{j|i}$ using random walks (proportion of random walks starting at i and reaching landmark j before any other landmark).

Learning a parametric embedding by preserving local structure

L. van der Maaten

Non-parametric dimension reduction techniques (Isomap, local linear embedding (LLE), Sammon, maximum variance unfolding (MVU)) cannot easily be extended out-of-sample. t -SNE can, by parametrizing the mapping from the (high-dimensional) input space to the (low-dimensional) output space using a deep feed-forward neural net (with layer dimensions n , 500, 500, 2000, 2), trained as auto-encoders are: pre-train each pair of consecutive layers as a restricted Boltzmann machine (RBM), using contrastive divergence (*i.e.*, minimize the Kullback-Leibler distance between the data and its reconstruction after one Gibbs step, instead of an infinity of steps (thermal equilibrium)), starting with the input layer, each RBM using the output of the previous as input; use the resulting weights as initial weights and use back-propagation (mini-batch) to optimize the t -SNE objective function.

Neural networks

G. Hinton (Coursera, 2012)

Neural networks were popular until the 1990s but went out of favour; thanks to the advent of more powerful computers (GPUs), deep neural networks can now be trained and the field is progressively merging with that of graphical models.

The course is a motley collection of neural net architectures (perceptron, feed-forward net, convolutional net, recurrent net, Hopfield net, restricted Boltzmann machine, deep belief net, auto-encoders) and related algorithms (back-propagation, biased heuristics, penalization, constraints (shared weights), careful weight initialization (echo state net), generative pre-training, momentum, bayesian methods, Gaussian processes), presented in a seemingly random order, preserved below.

1. A *neural net* is a set of *idealized neurons* (“units”), somehow linked together, to perform some machine learning task (supervised learning (regression, classification), unsupervised learning (e.g., as a first step before supervised or reinforcement learning, for dimension reduction), reinforcement learning). Expect neural networks to be bad at things the human brain is bad at (e.g., arithmetics). There are many types of idealized neurons:

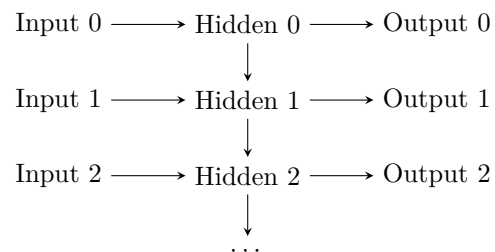
- Linear neuron: $y = b + \mathbf{w}'\mathbf{x}$;
- Binary threshold neuron: $y = \mathbf{1}_{b+\mathbf{w}'\mathbf{x} \geq 0}$;
- Rectified linear neuron: $y = (b + \mathbf{w}'\mathbf{x})_+$;
- Sigmoid neuron: $y = s(b + \mathbf{w}'\mathbf{x})$;
- Stochastic binary neuron: $y = 1$ with probability $s(b + \mathbf{w}'\mathbf{x})$, 0 otherwise.

In a *feed-forward neural network*, information always flows in the same direction, from the input layer, through the hidden layers, to the output layers (the neurons should be non-linear, otherwise it is equivalent to a net with no hidden layers).

Recurrent networks contain cycles; they are more powerful, but more difficult to train.

Input \longrightarrow Layer 1 \longleftrightarrow Layer 2 \longrightarrow Output

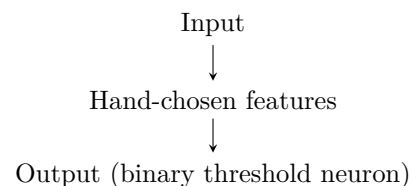
They are often unrolled and used to model sequences:



The weights do not depend on time; the hidden states can keep information for some time.

Symmetric neural networks are a special case of recurrent neural networks, where the weights are the same in each direction. They are more tractable. Teaching a symmetric net is equivalent to minimizing a simple function, called the “energy function”.

2. The most ancient neural net structure and learning algorithm is the **perceptron**.



The algorithm is straightforward: feed the data to the network; if the output is correct, do not change anything; if it outputs 0 instead of 1, add the input vector to the weights; if it outputs 1 instead of 0, subtract the output vector from the weights. If the data is linearly separable, it converges. But we need to find good features – ourselves.

3. Feed-forward neural networks with hidden units can discover relevant features themselves. They are usually trained with **back-propagation**, *i.e.*, by minimizing the quadratic error (the algorithm itself is just the chain rule for differentiation, expressed algorithmically: different types of neurons (transfer functions)

only change a multiplicative factor), with all the observations at a time (batch learning), or with one observation at a time (online learning) or, better, with groups of observations (mini-batch learning, to reduce zig-zags on the error landscape).

4. If the output is a non-binary discrete variable, you can use a *softmax*,

$$y_i = e^{z_i} / \sum_j e^{z_j}$$

(like a logistic function, but for more than two classes). For the cost function, you can use (minus) the logarithm of the probability of a correct answer or the cross-entropy, $-\sum \text{target}_i \times \log(\text{output}_i)$.

If this creates too many output neurons, you can add the discrete output variable to the inputs, and use a boolean output. (It is also possible to arrange the possible outputs in a binary tree, of which a single branch would be processed for each input.)

This can be used to model **relational data**, e.g., subject-verb-object triplets, using subject and verb as input, and the object as output: this can highlight regularities in a relational database, or find unlikely triplets (potential errors).

It also applies to language models: contrary to trigram models, which cannot understand similarities between words (e.g., dog/cat), the hidden layers (they can actually be learnt separately, and reused for other tasks) provide a representation of the context.

One can use dimension reduction, e.g., *t*-SNE (better than PCA or its non-linear analogues when there are clusters in the data) to represent the feature vectors in two dimensions.

5. In computer vision, to recognize objects, neural nets need to account for viewpoint invariance. This can be achieved by **convolutional neural nets**: each neuron in the hidden layer corresponds to (say) a 9×9 pixel area, but there are many similar neurons, constrained to have the same weights, to recognize the same feature in different positions. In the learning algorithm, the gradients are computed as usual, but modified to ensure that the constraints remain satisfied.

Good object recognition neural nets can have many (7) hidden layers, the firsts of which are convolutional.

6. **Learning** neural nets, especially deep ones, is tricky. The problems are often not caused by local minima, but by plateaus: for instance, if the learning rate is too high, the weights become so extreme that the output no longer depends on the inputs, and learning stalls. Classification problems also have a plateau on which the network only outputs the proportion of each class – learning useful features will take time, all the more so if the network is deep.

For small datasets (say 10,000 cases) standard optimization algorithms such as conjugate gradient (steepest descent in the direction of the gradient, then go in an "orthogonal" direction) or hessian-free methods (L-

BFGS uses a low-rank approximation of the hessian) are satisfactory. They can be combined with *stochastic gradient descent* (i.e., mini-batch learning).

Changing the *learning rate* may improve learning: if the error gets worse or oscillates, reduce it; if it decreases, slowly, increase it; at the end, turn down the learning rate, to smooth away the fluctuations due to the different samples. One can also use a separate learning rate for each parameter (e.g., a node-specific multiplicative gain, restricted to $[.1, 10]$ or $[.01, 100]$ applied to a global learning rate).

The *initial weights* are important: initialize them at random (otherwise, the units would all have the same weights: there would be no way to break the symmetry); and make their size proportional to the square root of their in-degree (otherwise, the output of neurons with a high in-degree has extreme values).

Preprocessing the input may also help learning: shift the inputs so that their average be zero, rescale them so that their variance be 1 (this makes the error surface more spherical); decorrelate the inputs, e.g., with principal component analysis (PCA).

Momentum can improve convergence and help escape local minima and plateaus: use the gradient to change the velocity, rather than the position; use a small momentum at the beginning (0.5), then raise to 0.9 or 0.99 – that is the most common method to learn large neural networks: mini-batches, stochastic gradient descent and momentum. While the usual momentum computes the gradient, then updates the direction, then jumps, one can first jump (in the previous direction), then compute the gradient, then update the direction and the position (Nesterov momentum).

Rprop only uses the direction of the gradient, not its amplitude, with a separate learning rate for each parameter. This only works with full batch learning: with mini-batches, use *rmsprop*, i.e., divide the gradient by a running average of the amplitude of the gradients.

7. Sequences are often modeled with autoregressive models, autoregressive models with hidden units (no memory), autoregressive models with hidden states and dynamics (linear dynamical system, aka Kalman filters, for continuous states, or hidden Markov models (HMM), for discrete states – but if there are n states, you can only store $\log n$ bits of information). **Recurrent neural networks** (RNN) are similar to HMM, but their hidden state is distributed, i.e., they have more memory, and they can describe non-linear dynamics.

For instance, adding two binary numbers of arbitrary length can be done with a 4-state finite automaton: it requires $\log_2 4$ bits of information, so we expect to find a RNN that solves this problem with fewer hidden neurons.

The most common task, predicting the next element of the sequence, blurs the difference between supervised and unsupervised learning.

After expanding a RNN over time, it is just a feed-forward network, with constraints on the weights: it can be trained by back-propagation through time.

8. Learning deep networks is problematic: the forward pass is fine, since the squashing (logistic) functions prevent the activities from exploding, but since the backward pass is linear (we replace the squashing functions with their derivative), it looks like A^n , where n is the number of layers – it will be either very large or very small. One can use an **echo state network** (ESN), *i.e.*, fix the input-hidden and hidden-hidden connections randomly (setting most of the weights to zero to have a sparse network, and rescaling them so that the amplitude of the activity vector remains the same), instead of learning them (it is a random equivalent of the kernel trick used by support vector machines (SVM)). This is sufficient for 1-dimensional time series, but for high-dimensional time series (video, sound), you can use the ESN as initial weights, and refine them with back-propagation (say, rmsprop with momentum).

One can also proceed as in electronics, and build RNN from modules, *e.g.*, “memory cells” able to remember things over a longer time.

9. The best cure against **overfitting** (learning the regularities specific to the training set – sampling error) is more data. If this is not possible, try a simpler model with few hidden layers.

Early stopping may help: start with small weights, and stop before overfitting (look at the performance on the validation dataset and stop when it gets worse) – the rationale is that a network with small weights is almost linear, therefore equivalent to a linear network with no hidden layers – the capacity will be sufficiently small.

You can limit the size of the weights, with an L^1 or L^2 *penalty*. With an L^2 penalty, if two weights play the same role, each will have half the weight (rather than, say, 1 and 0 or 101 and -100). With an L^1 penalty, you end up with many weights equal to 0; you can use other penalties, *e.g.*, $\text{Min}\{|x|, 1\}$, which penalizes small weights in the same way an L^1 penalty does (which leads to many zero weights), but does not penalize large weights. You can use weight constraints instead of weight penalties; you can put the constraint on a unit (say, L^2 norm of its weight vector), rather than on each individual weight.

Adding noise to the input of a linear net is equivalent to adding an L^2 penalty on the weights (adding an L^2 penalty is equivalent to adding a bayesian prior). You can add noise to the weights in recurrent nets. You can add noise to the activities, by using binary stochastic units in the forward pass, and logistic in the backward pass.

Averaging different models, either models with different forms, or models trained on different subsets of the data (bagging), improves performance. You could also consider a Bayesian approach: use a single neural net, but look at the posterior distribution of the weights, instead of just one set of values, and average

the predictions. For instance, the maximum a posteriori (MAP) estimator maximizes a penalized sum of squares, with the penalty being

$$\frac{\text{Var}(\text{noise})}{\text{Var}(\text{weights})} \times \sum \text{weight}_i^2.$$

You do not even have to assume the model is linear:

$$P[\text{Weights}|\text{Data}] = \frac{P[\text{Data}|\text{Weights}] \times P[\text{Weights}]}{P[\text{Data}]}$$

The first factor is unchanged, the second is the prior (the penalty), the last, a constant.

The *Empirical Bayes* cheats, and uses the data to decide on the prior: fit the model, look at the variance of the residuals and of the weights (*e.g.*, in each layer) use the ratio as the penalty coefficient, iterate.

Use cross-validation to choose the meta-parameters (or simply split the dataset in three: training data, validation data, test data).

10. There are many ways of **combining** multiple neural networks to improve generalization.

The error in a model can be decomposed into bias and variance. The bias is the inability of the model to learn the complexity of the phenomenon. The variance is the ability of the model to learn (undesired) sample-specific regularities. If you average the predictions of several models, the variance terms cancel out.

Bagging trains several models on different “subsets” of the data, obtained by sampling with replacement. For instance, bagging decision trees produces random forests (bagging neural networks would take a long time).

Boosting uses the same idea, but with weights instead of samples; the weights are not random, but focus on the misclassified cases: train a model on the data; then, train another model on the data, but give higher weights to test cases with a bad prediction; iterate.

Mixtures of experts combine several models, each trained on a part of the data (different regimes). A managing neural net looks at the input, infers the regime, and asks the corresponding expert. Since each expert only looks at a subset of the data, this requires a lot of data. This is not unlike boosting, but with boosting, the weights were only used in the training phase, not when using the network.

In the learning phase, one minimizes the squared distance between the target and the average (weighted) prediction (better, one could use the log-probability of the corresponding mixture of Gaussians), where the weights (competence of each expert for this problem) also have to be determined: each model tries to compensate the errors of the others, not to give a good prediction – this fosters cooperation.

Full bayesian learning does not output a single number, as the MAP estimator, but the full posterior distribution; you can use it to compute, say, the expectation of the forecasts. You can use complicated models with

little data – overfitting is the result of looking at a single number instead of the whole posterior distribution. The complexity of the model is a prior belief: it should not be influenced by the data... The posterior distribution can be estimated by putting a grid on the parameters and computing $p(W = w|\text{data})$ for all w on the grid.

To make full bayesian learning practical, you can use MCMC to sample from the posterior distribution, instead of a grid. Thanks to sampling noise, it works with mini-batches.

The *dropout* method is an approximation of the full bayesian approach, combining a very large number of networks, without having to train them all: for each test case, drop some of the units at random (with probability .5). Each model only has one test case, but the models share weights: this provides the needed regularization. This also prevents units from trying to fix the errors of other units instead of focusing on the task at hand. For prediction, use all the units, but halve their weights. For nets with one hidden layer, this gives the geometric mean of all the 2^H models. You can use the same idea on the input units, but with a high probability of keeping the units. Dropout reduces overfitting, better than early stopping.

11. A Hopfield net is a recurrent symmetric net with binary threshold units. Learning minimizes the “energy function”

$$\text{Energy} = - \sum_j s_i b_i - \sum_{ij} s_i s_j w_{ij}$$

where s_i is the state of unit i and w_{ij} the weight of edge $i-j$. Local minima of the energy function are memories; the network can store many memories, in a distributed way, and recover partial or corrupted memories. To increase the capacity of the network (the number of memories that can be stored – local minima close to one another tend to merge), *unlearn*: put the network in a random state, find the nearest local minimum, and learn its opposite – this is not unlike human sleep.

Hidden units in a Hopfield net encode an interpretation of the state of the visible units. For instance, in computer vision, visible units could be 2-dimensional lines on the sensor and hidden units 3-dimensional lines in space.

To escape from local minima, replace the binary threshold units with binary stochastic units,

$$p = \frac{1}{1 + e^{-\Delta E_i}}$$

or, even better,

$$p = \frac{1}{1 + e^{-\Delta E_i/T}}$$

i.e., simulated annealing. The network no longer converges to a single state, but to a probability distribution on the states, the “thermal equilibrium”, and the

probabilities are proportional to e^{-E} (Boltzman distribution); it can be sampled with Monte Carlo simulations.

A **Boltzman machine** is a stochastic Hopfield net with hidden units. They are good at modeling binary data, e.g., identifying the type of a text from a bag of words.

The Boltzmann learning algorithm maximizes the product of the probabilities of the vectors to learn. It turns out to be equivalent to:

$$\frac{d \log p(v)}{dw_{ij}} = \langle s_i, s_j \rangle_{\text{visible units clamped}} - \langle s_i, s_j \rangle_{\text{free}};$$

the two terms can be interpreted as learning and unlearning. To speed up the computation of those expectations (you have to wait until the machine settles in thermal equilibrium), start in the previous state, or “particle” (data-driven particles for the learning term, one for each data vector, fantasy particles for the unlearning term).

12. In a restricted Boltzman machine (RBM), there are no connections between hidden units or between visible units, *i.e.*, it is a bipartite network.

RBMs can be taught with *contrastive divergence*, alternating a learning and an unlearning phase (start with actual data, compute the hidden states, reconstruct the visible states (it looks like the initial data, with a few differences) and iterate many times, until thermal equilibrium: you end up with a completely distorted version of the initial states, which can be unlearned; this works well even if you do not wait until the thermal equilibrium: just take the first or second reconstruction (increase that number as the algorithm progresses)).

Collaborative filtering deals with a large matrix of ratings, with users in one dimension and items in the other, and most of the data is missing. Language models (modeling triples user-item-rating) convert each user into a feature vector, each item into a feature vector, and combines those vectors, e.g., with a scalar product, to produce a rating: this is actually a matrix *factorization model*, the most commonly used model for collaborative filtering. RBMs can tackle this problems: one visible unit per item, 100 hidden units, each training case corresponds to a user; to deal with missing values, discard the visible units with missing data: there are different networks, reasonably small, each with a single observation, but they share weights.

13. Back-propagation has a bad reputation: it does not work well with deep neural networks (those with many hidden layers, unless you impose a lot of constraints, as with convolutional nets), or with recurrent nets, and support vector machines (SVM) often perform better. The real problems are small datasets (neural networks need a lot of data), slow computers (to deal with the volume of data – GPUs are popular with neural network practitioners), simplistic networks (for simple models, low-dimensional datasets (up

to 100), or data plagued by noise rather than complex phenomena, statistics works best), and badly initialized weights.

A **belief net** (“Bayesian network”, in graphical model parlance) is a sparsely connected, directed acyclic graph, with stochastic variables. Contrary to Boltzmann machines (undirected graphical model, energy-based, “pairwise Markov networks” in graphical model parlance), belief nets are causal: they can easily be used to generate samples.

To initialize the weights carefully, to improve learning, one can use a generative model.

Learning sigmoid belief nets would be easy if we could sample from the posterior distribution of the hidden states. But, because of “explaining away”, it is difficult: if A and B are independent, improbable, and both imply C , they are negatively correlated given C – it is almost an exclusive or. MCMC can be used, but it is extremely slow: a Markov chain would get stuck on A or B and would never visit the other. *Variational methods* sample from an incorrect but simple approximation of the posterior.

The *wake-sleep* algorithm assumes that the posterior distribution (over the hidden configurations) factorizes into a product of distributions for each separate hidden unit, *i.e.*, that we can forget about explaining away. Consider two sets of weights: generative weights (the actual weights, from the hidden units to the observed ones) and recognition weights (in the other direction). In the wake phase, use the the data and the recognition weights, and sample the hidden units; given the state of those hidden units (considered as a sample from the posterior), train the generative weights. In the sleep phase, reverse the role of generative and recognition weights. The wake phase makes the recognition weights close to the (wrong) posterior; the sleep phase makes the wrong posterior close to the correct posterior.

14. RBMs are good at learning features, and those features can be reused as input of RBMs – and so on. Surprisingly, if you stack RBMs, the result looks like a sigmoid belief net – which suddenly becomes easy to learn.

Start by learning an RBM, with two layers, the data and a first hidden layer. Use this first hidden layer as input to another RBM: to learn it, you can initialize the second hidden layer with the data (the second RBM is just the first RBM upside down). You can then stack those two RBMs, to form a **deep belief net** (DBN) (it is a hybrid network, partly directed, partly undirected)

Hidden layer 2 \longleftrightarrow Hidden layer 1 \longrightarrow Data

The lower layers form a sigmoid belief net (perhaps with several hidden layers), the top two upper layers form an RBM. You may want to use different weights in the up and down phases.

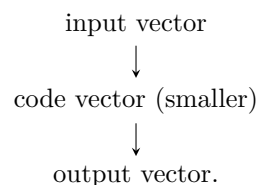
To generate data, use Gibbs sampling on the top layers (RBM), until equilibrium; then propagate towards the

visible layer, using the generative weights.

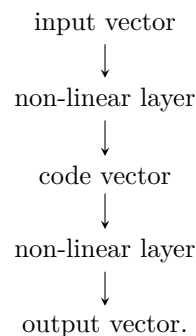
Stacks of RBMs can be used to initialize the weights before back-propagation: this pre-training is unsupervised (the labels are not used).

RBMs can also model real-valued data: use visible linear units with Gaussian noise and hidden rectified linear units.

15. Principal component analysis (PCA) can be implemented with back-propagation:



Use back-propagation to make the output vector as close as the input vector. This can be generalized with non-linear layers:



The first part is an encoder, the last a decoder – the whole is an **auto-encoder**.

Since back-propagation does not work well with deep networks, you can use echo-state-net weight initialization or unsupervised pre-training (use a stack of RBMs to initialize the encoding weights, use the transpose to initialize the decoding weights) before fine-tuning with back-propagation.

Latent semantic analysis, *i.e.*, PCA of the vectors of word counts (or probability vectors) is often used for document retrieval. You can use deep auto-encoders instead of PCA, *e.g.*, start with 2000 words, encode them to 500 neurons, then 250, then 10, and then decode them in the same way. Even with 2 dimensions, the plot is much more informative (the clusters are much more separated) than with (linear) PCA.

The **semantic hashing** problem is that of finding a document (text, image) similar to a given document: use an auto-encoder with logistic units in the code layer, add noise to the inputs of the code layer (otherwise, the logistic units would be used in their middle range, sensitive to noise, but conveying more information – instead, you can use binary stochastic units), and threshold them to get a binary code. Similar documents are documents with the same (or similar) bi-

nary codes (for similar documents, flip a few bits, *i.e.*, perform an exhaustive search on the neighbourhood).

Semantic hashing also works for image retrieval: first, use a small (28-bit) binary code to build a (long) shortlist of candidates. Then, use a 256-bit binary code, and perform an exhaustive search on the shortlist; the layers could be 1024 pixels, 8192 units, 4096, 2048, 1024, 512, 256 (learning may take days, even on a GPU).

Since an RBM is like a shallow auto-encoder, you can pre-train with a stack of auto-encoders instead of RBMs – it works better with *denoising auto-encoders*, *i.e.*, by adding noise to the input vector by setting many of the components to zero (like dropout, but for the inputs), or *contractive auto-encoders*, which make the activity of the hidden units less sensitive to the inputs by penalizing the squared gradient of the hidden activities wrt the inputs.

16. Gaussian processes can be used to optimize the hyper-parameters, in a Bayesian way: they estimate a function

$$f : \text{hyperparameters} \mapsto \text{quality of the result}$$

by providing an estimate and an error, *i.e.*, $f(x) \sim N(\hat{f}(x), \sigma_x)$, and one can evaluate the function at the hyperparameters with the biggest expected improvement on the best values so far

$$E[\text{improvement} | \text{improvement} > 0].$$

Probabilistic graphical models D. Koller (Coursera, 2012)

1. Knowledge representation

1a. Modelling a set of discrete random variables seems easy: you just have to give the joint probability table. However, when there are many variables, there are too many parameters to estimate – perhaps even, too many to store in the memory of your computer. **Graphical models** provide a parsimonious description of discrete probability distributions – think of them as “factor models” for discrete variables (a factor model is a parsimonious description of a covariance matrix, $V = \text{eve}' + \Delta$, with v small, positive definite, e rectangular, Δ diagonal and V large square, often used in finance to describe stock returns).

With the Bayes formula, you can decompose the joint probability

$$\begin{aligned} P(A, B, C, D, E) &= \\ P(A|B, C, D, E)P(B, C, D, E) &= \\ P(A|B, C, D, E)P(B|C, D, E)P(C, D, E) &= \\ P(A|B, C, D, E)P(B|C, D, E)P(C|D, E)P(D, E) &= \\ P(A|B, C, D, E)P(B|C, D, E)P(C|D, E)P(D|E)P(E). \end{aligned}$$

If you choose the order wisely, some simplifications can occur: for instance, if A is independent of C , D and E , you can replace $P(A|B, C, D, E)$ with $P(A|B)$. For

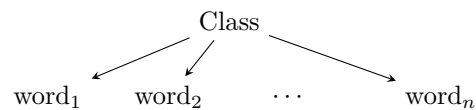
instance, we could have

$$p = P(A|B)P(B|D, E)P(C|E)P(D)P(E).$$

The decomposition can be described by a **directed acyclic graph** (DAG), with an arrow from x to y if $P(y|\dots x \dots)$ appears in the decomposition.

Conversely, a DAG defines a decomposition of the joint probability (to find the order: start with a leaf, remove it, iterate): a **Bayesian network** is a DAG with a *conditional probability distribution* (CPD) $P(X_i|\text{Parents}(X_i))$ at each node X_i .

The **naive Bayes model**, popularized by spam detection (or, more generally, document classification),



$$\text{word}_i \perp\!\!\!\perp \text{word}_j \mid \text{Class}$$

$$P(\text{Class}, \text{word}_1, \dots, \text{word}_n) = P(\text{Class}) \prod_i P(\text{word}_i | \text{Class})$$

can be the first step before a better-informed Bayes model. Inference is straightforward.

From a developer’s point of view, bayesian networks are easy to maintain (the model is just data) and can lead to more flexible user interfaces (since they can easily deal with missing (unobserved, latent) values, the user can choose not to answer a question).

Many bayesian networks are built from simpler building blocks (“ground networks” or templates):

- Dynamic bayesian models (hidden Markov models (HMM), Kalman filters, , all the chain-like models in NLP), temporal models (with the Markov assumption and time invariance, they can model arbitrarily long trajectories);
- Grid models in image processing (similar model for each pixel or superpixel);
- Object-relational model (e.g., pedigree relations in genetics; student–course relations in a university; the BUGS-like plate models, when there are many repeated elements).

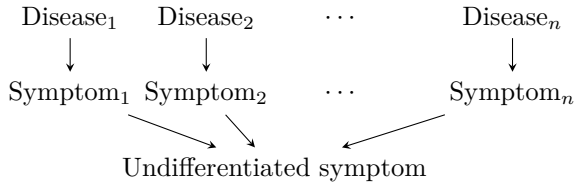
The distribution of $Y|X_1, \dots, X_n$ (say, when the X_i are binary) requires an exponential number of parameters: we often want a more concise description of those CPDs. **Tree-structured CPDs** encode context-specific independence

$$X \perp\!\!\!\perp_c Y | Z, C = c$$

(it just says that large parts of the table CPD are identical, and the decision tree is just a way of compressing it): the *multiplexer* CPD is one such tree

$$P(Y|K, X_1, \dots, X_n) = \mathbf{1}_{Y=X_K}.$$

In the **noisy OR**, each possible cause (say, diseases) triggers the consequence (a symptom) with some probability, and the consequence appears if any of the causes triggered it: this ignores dependence between causes.



Instead of OR, you could use AND, MAX, or a sum (sigmoid CPD, as in neural networks)

$$P(Y = 1|X_1, \dots, X_n) = \text{sigmoid} \left(w_0 + \sum_i w_i X_i \right).$$

1b. Markov networks are undirected graphical models. A **pairwise Markov network** decomposes the joint probability distribution as a product of “factors” (or “potentials”), one for each edge, e.g.,

$$P(A, B, C, D) \propto \phi_1(A, B)\phi_2(B, C)\phi_3(C, D)\phi_4(A, D);$$

for instance, the pixels (or superpixels) in an image form a lattice (an undirected graph). The factors do not have any direct interpretation. **Markov networks** (MN) (sometimes called **Markov random fields** (MRF) or general Gibbs distribution) do the same thing with hypergraphs.

$$P(X_1, \dots, X_n) \propto \prod_i \phi_i(D_i),$$

where the D_i are hyperedges. A Markov network can also be represented as a graph, with an edge between two vertices if they are in the same hyperedge. By replacing the hyperedges with their cliques, we lose some information.

1c. Conditional Random Fields (CRF) do not model $P(Y, X)$ but $P(Y|X)$: we can then ignore the distribution of X (the correlations do not matter – they could be difficult to model). Only the normalizing constant changes.

$$\begin{aligned} P(X, Y) &\propto \prod f_i(D_i) \\ Z(X) &= \sum_Y P(X, Y) \\ P(Y|X) &= P(X, Y)/Z(X). \end{aligned}$$

Log-linear models (feature-based models) are of the form

$$\begin{aligned} P &\propto \prod \phi_i(D_i) \\ \phi_i(D_i) &= \exp(w_i f_i(D_i)) \end{aligned}$$

where f_i is known (“feature”) and the w_i are parameters (there can be several factors with the same domain,

and you can impose that the weights be equal). Often, the features f_i are indicator functions, as in the Ising model. For metric MRF,

$$\begin{aligned} f_{ij}(X_i, X_j) &= \text{distance}(X_i, X_j) \\ \phi_{ij}(X_i, X_j) &= \exp(-w_{ij} f_{ij}(X_i, X_j)) \end{aligned}$$

where the distance can be a Kronecker delta, the absolute value of the difference, or some robust cost function such as $d(x, y) = \text{Min}(1, |x - y|)$.

BN are preferable when there is a natural direction (e.g., a causal network), MRF when there is none (e.g., adjacent pixels in an image), CRF when there are variables with a complicated dependency structure you do not want to model (e.g., complex features, in image analysis).

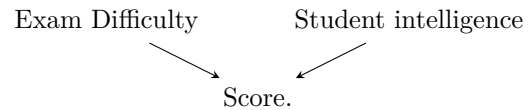
1d. The notions of **conditional independence**, causal reasoning (the path from evidence to conclusion follows the edge directions), evidential reasoning (opposite), intercausal reasoning (from one cause of an effect to another: $P(\text{Cause}_1|\text{Effect}, \text{Cause}_2)$, explaining away) and, more generally, inference flow can be expressed in graph-theoretic terms. Inference can flow from X to Y if they are linked by edges such as

$$\begin{aligned} X &\longrightarrow Y && \text{(causal relation)} \\ X &\longleftarrow Y && \text{(deduction)} \\ X &\longrightarrow W \longrightarrow Y \\ X &\longleftarrow W \longleftarrow Y \\ X &\longleftarrow W \longrightarrow Y && \text{(intercausal relation)} \end{aligned}$$

but inference does not flow through **v-structures**

$$X \longrightarrow Y \longleftarrow Z$$

for instance,



Trails with no v-structures, are called **active trails**.

Things are slightly more complicated with conditional inference: a trail $X_1 \longrightarrow \dots \longrightarrow X_n$ is active given Z if, for each v-structure $\longrightarrow X_i \longleftarrow$ it contains, X_i or one of its descendants is in Z , and no other X_i is in Z . It is simpler with undirected graphs: a trail $X_1 - \dots - X_n$ is active given Z if no X_i is in Z .

Bayesian networks can be interpreted as

- a factorization of the joint probability;
- a description of the dependencies among the variables;
- a description of the conditional independence relations between the variables.

The following properties are equivalent.

- $P \models (X \perp\!\!\!\perp Y|Z)$
- $P(X, Y|Z) = P(X|Z)P(Y|Z)$
- $P(X|Y, Z) = P(X|Z)$
- $P(Y|X, Z) = P(Y|Z)$

- $P(X, Y, Z) \propto \phi_1(X, Z)\phi_2(Y, Z)$ for some functions (“factors”) ϕ_1 and ϕ_2 .

If there is no active trail between X and Y given Z , then X and Z are said to be **d-separated** given Z and they are independent given Z . For instance, any node is d-separated from its non-descendants given its parents.

The **independence map** of a directed graph G is the set $I(G)$ of conditional independence relations coming from d-separation. A set of independence relations satisfied by a probability distribution P is said to be an **independence map** of P . One can show that a probability distribution P factorizes over a directed graph G iff $I(G)$ is an independence map of P .

For Markov networks as well, a probability distribution P factorizes over an undirected graph G iff G is an I-map for P (at least, if the distribution is positive, *i.e.*, never zero – in particular, there are no deterministic relations).

A graph is a minimal I-map if you cannot remove any edge – minimal I-maps exist, but need not be unique. A graph G is a perfect map if $I(G) = I(P)$ – perfect maps need not exist. Two graphs are I-equivalent if they encode the same independencies: there is an unidentifiability problem if you try to infer the structure of the graph.

It is possible to convert between BN and MN, but some independencies are lost: from BN to MN, we lose the independencies in the v-structures, from MN to BN, we must add triangulating edges to loops.

2. Inference

We have a network, some data (the evidence), and a question about it. Unfortunately, most inference problems on graphical networks are NP-hard.

A **conditional probability query** is the problem of computing $P(Y)$ (marginal probability) or, more generally, $P(Y|E = e)$ (*i.e.*, “reduce” the factors by the evidence). It is also called a **sum-product problem**: the aim is to compute an (exponentially large) sum of products. In the presence of evidence, or for undirected graphs, you also need to renormalize the result.

The algorithms include: variable elimination (exact), belief propagation (biased), variational approximation, MCMC (often with importance sampling).

The MAP (maximum a posteriori) inference problem is the problem of finding

$$\text{MAP}(Y|E = e) = \underset{y}{\text{Argmax}} P(Y = y|E = e)$$

where Y contains all the variables not in the evidence E . This is different from the maximum over the marginals. It is also called a **max-product problem**.

The algorithms include variable elimination, belief propagation, integer programming (efficient, popular), graph-cut methods (in some special cases), combinatorial search.

2a. Variable elimination (VE) integrates out one variable at a time from the sum-product; a judiciously-chosen elimination order can reduce the number of operations.

Variable elimination can be visualized as operations on a graph. First convert the graph to an undirected graph (you need to add edges for all v-structures, *i.e.*, create the cliques associated to the factors); then eliminate the nodes one by one (you need to add *fill edges* between the neighbours of the node you remove). We want an elimination order for which the induced graph (with all the fill edges added) has small cliques.

To find a good elimination order, one can use a greedy approach, *i.e.*, eliminate the node with the smallest cost, where the cost is one of

- the number of neighbours in the current graph;
- the number of values in the factor formed (if the factors are not all binary) (min-weight);
- the number of new fill edges (min-fill – it works much better);
- the total weight (number of values) of the new fill edges (weighted min-fill).

To find a good elimination order, one can look for a low-width (small cliques) triangulation (no loop of length greater than 3 without a “bridge”) of the initial graph (the induced graph is triangulated).

2b. Belief propagation (BP) considers a covering of the graph by subgraphs, which exchange information. A *cluster graph*, associated to a Markov network, is a graph whose vertices (clusters) are sets of variables, with edges labeled by sets of variables (sepsets) common to both clusters (not necessarily all the variables in the intersection), and each factor assigned to a cluster with all the required variables. The factor ψ of a cluster is the product of the (MN) factors ϕ associated to it. Clusters then exchange messages $\delta_{i \rightarrow j}$: initialize the messages to 1, multiply the messages you received with the information you have, and send it (suitably marginalized) to the other neighbours; when sending to a neighbour, do not use the information you directly received from it. After some time, compute the beliefs: $\beta = \delta\psi$. They often converge (but sometimes oscillate), and are biased (significantly, but for few variables – BP is useful when there are many variables). Convergence and accuracy are worse in presence of tight loops or potentials pulling in opposite directions.

Cluster graphs should satisfy the running intersection property: if two clusters have a variable in common, there is a unique path between them all of whose sepsets contain the variable. In other words, for any variable X , the clusters and sepsets containing it form a tree (connected, no loops).

The *Bethe cluster graph* is a bipartite cluster graph, with a cluster for each variable and for each factor, and edges between a variable and a factor if it is in its scope. It is a good first choice.

A cluster graph is said to be calibrated if clusters agree

on their sepsets.

Even if the beliefs are biased (as estimators of marginal probabilities), information is not lost: the unnormalized joint probability can be recovered as

$$\frac{\prod_i \beta_i}{\prod_{i,j} \mu_{i,j}}$$

where $\mu_{i,j} = \delta_{i \rightarrow j} \delta_{j \rightarrow i}$ are the sepset beliefs.

For *clique trees*, *i.e.*, cluster graphs that form a tree, and in which the sepsets are exactly cluster intersections, the belief propagation computations are equivalent to variable elimination and lead to correct marginals. Messages from a leaf are final: if you pass the messages from the leaves inwards (find a node that has received messages from all its neighbours except one), belief propagation converges immediately.

You can easily compute all marginals (with variable elimination, you had to redo everything for each marginal). To answer queries, you may need to recompute some messages, but not all: only those from the new evidence to the variable of interest.

Clique trees can have large clusters: each sepset separates the graph into two conditionally independent parts (this is a consequence of the running intersection property).

Do not use synchronous BP – asynchronous BP converges more often and faster. On an arbitrary cluster graph, propagation order matters:

- Tree reparametrization (TRP): choose a tree at random (it works better with larger trees, *e.g.*, spanning trees), calibrate it, and iterate with other trees (make sure you sample all edges);
- Residual belief propagation (RBP): pass messages between clusters whose belief over their common sepset disagree most (use a priority queue).

To limit oscillations, you can smooth (dampen) the messages (bias them towards their previous value).

Turbo codes (efficient error-correcting codes) were one of the flagship applications of loopy BP.

The MAP estimation problem is a Max-Sum problem, formally identical to the sum-product problem: it can be solved via variable elimination. Message passing can be used, but the beliefs at each clique are just max-marginals; however if the global MAP assignment is unique, it can be recovered from the max-marginals (if it is not, perturb the problem to make it unique, or build the assignment one variable at a time, backtracking if needed).

Some MAP problems are tractable (via ad hoc algorithms): matching (over a bipartite graph); associative potentials (when two binary variables are dependent, they tend to agree more often than not); metric MRF (a continuous analogue of associative potentials); cardinality factors ($\text{score}(X_1, \dots, X_n) = f(\sum X_i)$, where the X_i are binary); sparse pattern factors (the table CPD is sparse); convexity factors (contiguity in texts,

contiguous activities in temporal data, image segmentation), etc.

The **dual decomposition** uses optimization techniques to address the MAP problem: separate 1-variable factors from many-variable factors

$$\begin{aligned} \text{MAP}(\theta) &= \text{Max}_x \left(\sum_i \theta_i(x_i) + \sum_F \theta_F(x_F) \right) \\ &= \text{Max}_x \left(\sum_i \theta_i(x_i) + \sum_{F \ni i} \lambda_{Fi}(x_i) + \right. \\ &\quad \left. \sum_F \theta_F(x_F) - \sum_{i \in F} \lambda_{Fi}(x_i) \right) \\ &\leq \sum_i \text{Max}_x \left(\theta_i(x_i) + \sum_{F \ni i} \lambda_{Fi}(x_i) \right) + \\ &\quad \sum_F \text{Max}_x \left(\theta_F(x_F) - \sum_{i \in F} \lambda_{Fi}(x_i) \right) \end{aligned}$$

and solve those subproblems (or “slaves” – choose the decomposition so that they are tractable); when they disagree, *i.e.*, $x_{Fi}^* \neq x_i^*$, reduce the fitness on both sides

$$\begin{aligned} \lambda_{Fi}(x_i^*) &= \alpha \\ \lambda_{Fi}(x_{Fi}^*) &= \alpha; \end{aligned}$$

the algorithm converges if $\sum \alpha_t = \infty$ and $\sum \alpha_t^2 < \infty$. If the slaves agree, this is the MAP; if not, use heuristics (voting, decompositions into a spanning tree (it will be consistent), etc.) to “decode” them.

2d. Monte Carlo methods (Gibbs Sampling, Metropolis-Hastings) can also be used for inference.

2e. Graphical models are also used to support decision making: we do not only want to estimate a model, we want to find a decision rule that maximizes some expected utility (MEU). An *influence diagram* is a bayesian network with added “action” nodes and a “utility” node (if the utility function is separable *i.e.*, a sum of simpler functions, There can be several utility nodes).

This diagram can help decide which variable to measure, by computing the *value of perfect information*, the difference between the MEU of the influence diagram with an edge added (from the variable to measure to the action node) and the MEU of the initial influence diagram.

3. Learning. Given data and a partial description of the network (*e.g.*, the structure but not the parameters, or just the nodes, without the edges – in addition, some of the data can be missing, and some of the nodes may be unknown (latent variables)), we want a complete description of the graphical model (structure and parameters).

3a. Maximum likelihood estimation (MLE) is the most straightforward: the likelihood can be decomposed into a product of local likelihoods – but use cross-validation to prevent overfitting (in particular, the likelihood of a complete graph is always higher).

Learning is a special case of inference: parameters can be seen as random variables. *Bayesian learning* combines an imaginary Dirichlet sample (the Dirichlet prior generalizes the Beta prior, for the multinomial distribution instead of Bernoulli), with real data samples; it is asymptotically equivalent to the MLE (the relative size of the prior decreases as more data arrives), but converges faster.

One can also use maximum a posteriori (MAP) estimators.

3b. Structure learning is trickier. If an arc is missing, you cannot recover the true distribution, but the network may have good generalization properties. If there is an arc in excess, you can recover the true distribution, but the network may have poor generalization properties.

To learn the structure, one can define some score function (distance between model and data) and optimize it: consider all the possible graphs on the nodes you have, compute a score for each of them, and pick the graph with the highest score.

The *likelihood* is not a good scoring function: it is guaranteed to overfit – it prefers the fully-connected graph. You can add a constraint or, better, a penalty on the complexity of the graph, as in the *Bayesian information criterion* (BIC).

$$\text{log-likelihood} - \frac{\log(\text{number of instances})}{2 \times \text{number of independent parameters}}$$

(the opposite is sometimes called the MDL (minimum description length) criterion). It is asymptotically consistent.

The *bayesian score* is

$$P(\text{model}|\text{data}) \propto P(\text{data}|\text{model})P(\text{model});$$

it is dominated by the marginal likelihood $P(\text{data}|\text{model})$. This is not the likelihood but the marginal likelihood: the likelihood is evaluated at the best parameters, while the *marginal likelihood* is averaged over all possible parameters (for a given graph structure). The marginal likelihood can be easily computed for multinomial models: it is a sum, one term for each node, only involving information about its parents. For the structure prior, $P(\text{model})$, you can use a constant (uniform distribution on the space of graphs: the marginal likelihood is penalized enough), or a penalty for the number of edges or parameters. For the parameter priors (needed to compute the integral inside $P(\text{data}|\text{model})$), use a Dirichlet prior, *i.e.*, the parameters for a network with no edges, or the parameters from a simple network structure. Asymptotically, the Bayesian score is equivalent to the BIC score: in particular, it is consistent.

The scoring function is decomposable: it is a sum over the nodes (or “families”: each node and its parents). In the case of trees or forests, it can be transformed into a sum over the (undirected) edges, and finding

the optimal tree (or forest) is just a minimum spanning tree problem. For more general graphs, the problem can still be reduced to a graph-theoretic one, but it is NP-hard: we have to resort to some heuristic, such as greedy hill climbing or simulated annealing – to avoid local maxima and plateaux (equivalent networks are often neighbours), use random restarts and a tabu list, and make sure the neighbourhoods are sufficiently large (in particular, you do need edge reversals, otherwise there are many more local extrema).

The decomposability can help speed up the implementation: we only need the differences in scores, which only involve one or two nodes, and do not change unless we touch those nodes: just put the possible moves in a priority queue (heap) sorted by the score differences.

3c. Hidden variables can simplify the structure of the network, but make the optimization problem more complex: because of unidentifiability (e.g., values of a discrete hidden variable can be permuted), which leads to the multiplication of local extrema, and non-decomposability. Missing data pose the same problem. One could use gradient descent: there is a simple formula for the gradient, but it requires inference for each data instance. Expectation maximization (EM) is faster: it uses the fact that given complete data, parameter estimation (M step) is easy; and that given parameters, computing the distribution of missing data (inference, E step – it is a *soft completion* of the data: not a single value but a probability distribution) is easy (uses the same efficient statistics as complete data MLE). You should not only look at the likelihood, but also at the parameter values (they continue to fluctuate even though the likelihood seems to be stable); you should use cross-validation to know when to stop the optimization (the longer it runs, the higher the overfitting risk); you should use several restarts (random, or from prior knowledge, or from a simpler (clustering) algorithm).

Successful uses of probabilistic graphical models mix and match models (directed and undirected), inference algorithms (approximate (BP, MCMC) and exact on some subsets of variables), and learning algorithms (e.g., SVM for some potentials, CRF for others).

Market Risk Analysis IV Value-at-Risk Models C. Alexander (2008)

1. There are many measures of risk: standard deviation, semi-variance, lower partial moments, etc.. The **value at risk** (VaR) is a quantile of the distribution of discounted returns, for a given significance level α and horizon h . It is not a *coherent* risk measure, but once you have a VaR model, you can also compute the *expected shortfall* (ES). The VaR (or ES) can be measured absolutely or with respect to a benchmark.

The expected returns can be neglected in short-term VaR, but not in very long-term VaR. For long-term

VaR, the difference between *dynamic VaR* (continuously rebalance the portfolio to keep the risk exposures constant) and *static VaR* (do not touch the portfolio, let it drift) becomes more pronounced.

The greeks are sometimes incorrectly seen as risk measures: they are only *risk sensitivities* – the actual risk is the product with the risk factor returns.

2. Equity and cash flow portfolios are linear in their risk factors (strictly speaking, cash flow portfolios are linear in the discount factors, not the rates, but the linear approximation is very good). When there is a term structure of risk factors, principal component analysis (PCA) can reduce their number.

The **parametric linear VaR** model is not limited to Gaussian iid risk factor returns: it can be corrected for autocorrelation (AR(1)); it can be generalized to a Student distribution (with heavy tails) or a mixture distribution (to model different regimes).

To account for volatility clustering, one can estimate the variance matrix with an exponentially weighted moving average (EWMA) or, better, a GARCH model.

Model risk can be very high: try several models.

3. Historical simulation, *i.e.*, the use of historical returns (for the factors or for the portfolio), provides VaR estimates with no distributional assumptions. The returns should be adjusted so that the volatility be the same as the current volatility:

$$y_t = \frac{\hat{\sigma}_T}{\hat{\sigma}_t} x_t,$$

where $\hat{\sigma}$ comes from a GARCH (or asymmetric GARCH (A-GARCH), to account for the leverage effect) model. In a multivariate setup, divide and multiply by the Choleski matrices instead of the volatilities.

The square root rule, for *time scaling*, is only valid for Gaussian iid returns. For stable iid returns, there is a similar rule, with a different exponent (to be estimated). For non-iid returns, you can use an h -period ahead volatility forecast from a GARCH model.

To estimate extreme quantiles from historical data, you need to model the tail, *e.g.*, with kernel density estimators; extreme value theory (EVT) (but you have to choose a threshold); the Cornish-Fisher expansion (which tends to over-estimate the VaR); or (better) the Johnson SU distribution.

To decompose the VaR, decompose the returns

$$\text{return} = \sum_i \beta_i \times \text{return}_i + \varepsilon_i.$$

One can compute the VaR of each term (*standalone VaR*, not additive), the VaR from all the risk factors (*systematic VaR*) or from the residuals (*specific VaR*), or use Euler's identity and numeric differentiation to have an actual decomposition (*marginal VaR*, *incremental VaR*). For instance, for FX risk, you can use

$$\text{log-returns}_{\text{GBP}} = \text{log-returns}_{\text{USD}} + \text{FX log-returns}$$

(you may need to assume that log-returns and ratio-returns are the same, which may be fine for very short horizons).

4. Monte Carlo simulations are often preferable to linear VaR or historical VaR models: just sample from a statistical model for the risk factor returns.

GARCH (or, better, *Markov-switching GARCH*) models can account for volatility clustering. Principal component analysis (PCA) will reduce the number of factors and make them orthogonal. *Copulas* can model the dependence between risk factors: even elliptic copulas as useful, since they allow different marginal distributions (*e.g.*, Student with different degrees of freedom, or Gaussian mixtures).

Importance sampling can be used to improve the precision (reduce the *sampling error*) of the VaR – but it is a quantile, not an expectation: the details are different.

5. The main risk factors for **option portfolios** are price, squared price (to account for non-linearities) and implied volatility. The *value delta* (or dollar delta) is

$$\Delta_{\$} = \frac{\partial \text{Price}}{\partial \log S} = S \frac{\partial \text{Price}}{\partial S} = S \Delta.$$

It can be aggregated over different underlyers – it answers the question “how much would the value of the portfolio change if all prices simultaneously rose by 1%”. The other value greeks are defined similarly, and may be used to compute the Δ - Γ approximation

$$\text{P\&L} = \Delta'_{\$} R + \frac{1}{2} R' \Gamma_{\$} R,$$

where R is the vector of discounted returns for the underlyers.

The corresponding VaR can be computed analytically: one can compute the moments of the P&L distribution and either use the Cornish-Fisher expansion or (better) fit a Johnson SU distribution.

Time scaling may look straightforward if you only look at prices

$$\text{P\&L}_h = h^{1/2} \Delta'_{\$} R + \frac{1}{2} h R' \Gamma_{\$} R,$$

but it is not clear how to scale vega (the implied volatility is mean-reverting) – it is preferable to directly use h -day returns.

While the Δ - Γ -*vega approximation* (the other greeks do not add any significant information) is fast and often used for intraday (real-time) VaR, it is inadequate, especially for hedged portfolios: it is preferable to exactly reprice the options, using either historical data, or a dynamic model for the risk factors (volatility clustering, mean reversion for risk indices and implied volatility, etc.).

If there are too many risk factors (say, futures of different maturities on various underlyers; implied volatilities for many strikes, maturities and underlyers), you can use principal component analysis (PCA) to reduce their number and identify the “main” ones. PCA does not work well with implied volatilities: instead, you can

use “volatility beta mapping”, *i.e.*, choose a reference volatility (usually ATM).

As usual with options, one should not confuse the **P** and **Q** measures: the VaR is a quantile of the real-world (**P**) distribution of returns.

6. There are many sources of **model risk**:

- Volatility clustering, autocorrelation;
- Heavy-tailed return distributions;
- Choice of risk factors (e.g., vertex choice, for cash flow portfolios);
- Estimation procedures (e.g., the “ghost artefacts” in rolling unweighted OLS estimates: the estimators jump when an outlier enters and leaves the window);
- Inadequate scaling (the regulators encourage this: they ask banks to use a 10-day VaR to compute their capital requirement, but only mandate tests on the 1-day VaR, and allow the square root rule...).

You can compute confidence intervals for the VaR estimators (analytically, or via asymptotic results – bootstrap is not mentioned). Here are a few tests for the VaR, based on the number of *exceedances* n_1 and the number of consecutive exceedances n_{11} (the tests are only asymptotic, unlikely to be useful unless you really have a huge amount of data, and the independence test will overlook clustering unless the exceedances are exactly consecutive).

$$\begin{aligned} -2 \ln \frac{\alpha^{n_1} (1 - \alpha)^{n_0}}{\left(\frac{n_1}{n}\right)^{n_1} \left(\frac{n_0}{n}\right)^{n_0}} &\sim \chi^2_{\text{df}=1} \\ -2 \ln \frac{\left(\frac{n_1}{n}\right)^{n_1} \left(\frac{n_0}{n}\right)^{n_0}}{\left(\frac{n_{00}}{n_0}\right)^{n_{00}} \left(\frac{n_{01}}{n_0}\right)^{n_{01}} \left(\frac{n_{10}}{n_1}\right)^{n_{10}} \left(\frac{n_{11}}{n_1}\right)^{n_{11}}} &\sim \chi^2_{\text{df}=1} \\ -2 \ln \frac{\alpha^{n_1} (1 - \alpha)^{n_0}}{\left(\frac{n_{00}}{n_0}\right)^{n_{00}} \left(\frac{n_{01}}{n_0}\right)^{n_{01}} \left(\frac{n_{10}}{n_1}\right)^{n_{10}} \left(\frac{n_{11}}{n_1}\right)^{n_{11}}} &\sim \chi^2_{\text{df}=2} \end{aligned}$$

There are similar tests for the expected shortfall.

To check if the risk model uses all the information in the market, regress the exceedance indicator variable against the lagged risk factors (with several lags) and compare with the intercept-only model (the intercept is the significance level α).

If you are willing to assume Gaussian iid returns (test for autocorrelation, heteroskedasticity, normality), you could test the bias:

$$\text{sd} \left(\frac{Y_{t+1}}{\hat{\sigma}_{t+1}} \right) \sim N \left(1, \frac{1}{2T} \right).$$

Instead of looking at a single quantile, you could compare the whole forecasted distribution with the actual returns, e.g., with a likelihood ratio test (I would use a Kolmogorov-Smirnov test).

7. While VaR and ES look at extreme events in normal market conditions, **stress tests** (scenario analysis) look at abnormal market conditions.

Some *stress tests* are computed from the risk factors returns for a historical event (1987 crash, etc.).

Some rely on single-case scenarios, *i.e.*, a single vector of risk factor returns: for instance the *worst-case loss* stress tests look at the worst possible loss if the risk factors move by $\pm 6\sigma$ – but, since the portfolio may be hedged along those risk factors, it may be preferable to look at the losses for large changes in the principal components (especially if there is a term structure of risk factors) or if the risk factor returns are on the ellipse $x'\Omega^{-1}x = c$ (this uses the risk model Ω) – for non-linear portfolios, you should even look inside this ellipse (“trust region worst case loss”).

Other stress tests rely on distribution scenarios: for instance, the conditional distribution of the risk factor returns given that one of them jumped, $(f_j)_{j \neq i} | f_i > a$; or some subjective view; or a mixture distribution (the normal regime, from the risk model, and a “crash regime”, with some crash probability); or a bayesian blend of the two regimes (product of the probability distribution functions).

You can use data from a past crisis (sample covariance matrix, higher moments) to define the “crash regime” or correct the covariance matrix of (more recent) historical returns. If you manually tweak the correlation matrix, it can cease to be a valid correlation matrix, and you may need to correct it.

Exogenous *liquidity adjustment* of the VaR (by modeling the distribution of the bid-ask spread) has a negligible effect, but endogenous adjustment (linear or quadratic market impact, with some price drift if the liquidation is spread over several days, accounting for volatility clustering) may be necessary.

8. Regulatory capital is the capital that regulators ask banks to hold to ensure they remain solvent. It can be computed as $3\text{VaR}_{10\text{-day}}$ (either the average of the 10-day VaR over the past 60 days, or the latest 10-day VaR, whichever is higher) if the VaR model has been approved (*i.e.*, if it does not have too many exceedances). Alternatively, the regulators provide “standardized rules”: for equities, 8% of the value of the positions, plus a 4% to 8% charge for specific risk (insufficient diversification), and another one for credit risk – fixed income, commodities, currencies have similar rules.

Economic capital (EC) is similar, but used internally and presented to shareholders and rating agencies. It is often based on ES rather than VaR. *Risk budgeting* or *economic capital allocation* is the allocation of the economic capital to the various desks of the firm. Contrary to real capital (needed by funded activities, and whose financing cost should be included in the P&L), it is not additive, because of diversification. *Aggregation risk* (use of incorrect correlation when aggregating the VaR or ES of the various desks to compute the firm-wide economic capital, or when allocating the economic capital to the various desks – or incorrect use of correlations when they do not capture the dependences) is the

most important source of model risk. Risk budgeting is often done by looking at the efficient frontier in the $P\&L \times EC$ space, or by optimizing some risk-adjusted performance measure, such as

$$RORAC = \frac{E[P\&L]}{EC}$$

or

$$RAROC = \frac{E[P\&L] - kEC}{EC}$$

(they give the same point on the efficient frontier: the tangent allocation).

Social network analysis **L. Adamic (Coursera, 2012)**

1. Given a network, you can look at the following *descriptive statistics*.

- The degree of each node, the distribution of the degrees (distinguish in- and out-degrees for oriented graphs), the average degree, its standard deviation or the Gini coefficient (to measure how diverse the degrees are);
- The average shortest path;
- The size of the largest connected component (or strongly-connected, for oriented graphs).

To assess how central a node is, look at:

- Its degree;
- Its *betweenness centrality*, i.e., the proportion of pairs of nodes on whose shortest path the node is;
- Its *closeness centrality*, i.e., the average of the inverse of the distances from this node;
- Its *eigenvalue centrality*,

$$\text{centrality}_i = \sum_{j \text{ neighbour of } i} w_{ij} \text{centrality}_j,$$

which can be generalized to $C = (\alpha + \beta C)A$, where A is the adjacency matrix, β a parameter, and α a normalization constant. The analogue for directed networks, PageRank, is more complicated: it can be computed with a random walk on the directed graph, with some teleportation to avoid getting stuck in loops (it cannot happen with undirected networks: if you enter a loop, you can always leave it).

The strength of an edge A–B can be assessed with the *neighbourhood overlap*:

$$\frac{\text{number of neighbours of both A and B}}{\text{number of neighbours of either A or B}}.$$

The presence of communities can be assessed by the *clustering coefficient*:

$$\frac{\text{number of closed triangles}}{\text{number of connected triplets}}.$$

For a directed network, you can count all the 3-node patterns (“motifs”) and compare with the numbers for a random graph with the same average degree, or with the same degree distribution.

Identifying communities is trickier (most of those algorithms do not work on real-world networks, probably because communities overlap):

- Connected components, strongly-connected components;
- Cliques: they betray community structures, but they overlap, can be incomplete, are not robust; they only describe a “core” rather than the larger community;
- k -core: a subgraph in which every node is connected to at least k other nodes;
- Connected components of the graph with nodes the k -cliques and an edge if two k -cliques differ by only one node;
- Hierarchical clustering;
- Betweenness clustering: remove the edge with the highest betweenness, recompute, iterate until the betweenness of the remaining edges is sufficiently low; the connected components are the communities;
- Modularity: compare the presence of edges within and between communities with what you would expect if they were selected at random; start with random community assignments and change them to improve the modularity.

2. The simplest network model is the *Erdős-Rényi* random network: the edges are simply selected at random, independently. There is a giant component, there are no hubs, the degree distribution is binomial (asymptotically Poisson), the average shortest path is proportional to $\log n$.

Growth models, such as the *Barabasi-Albert preferential attachment model*, have hubs, and a more unequal degree distribution (power law).

Small world networks combine the short distances of ER random networks with a community structure. The *Watts-Strogatz model* starts with a lattice, starts to randomly rewire it, but stops before getting a random network. There are hierarchical (Watts-Dodds-Newman) and geographical (Kleinberg, the probability of an edge is proportional to some power of the distance). When rewiring, you can use simulated annealing to minimize

$$\lambda \times \text{average shortest path, in number of edges} + (1 - \lambda) \times \text{average shortest path, in kilometers}.$$

Changing the value of λ gives networks with hubs or with short edges

3. A *power law* (or Pareto distribution) is a probability distribution such that

$$P(X = x) \propto x^{-\alpha};$$

in the case of node degree, the variable is discrete. The *Zipf law* (the size of the r th largest event is proportional to $r^{-\beta}$) is the q-function of the Pareto distribution. Fitting a (discrete) power law distribution is tricky:

- When regressing the logarithm of the number of nodes with a given degree against the logarithm of

the degree, large numbers of observations are binned in a single point for low degrees, and there are many empty bins for high degrees – forgetting them gives a very biased estimator;

- Logarithmic binning is better, but discards information;
- Cumulative binning is preferable: the number of events in $[1, n]$ still follows a power law, with exponent $\alpha - 1$;
- The maximum likelihood estimator is (you need to choose where the power law starts, x_{\min})

$$\alpha = 1 + n \left(\sum \log(x_i/x_{\min}) \right)^{-1}$$

- The power law may not extend to the very end of the tail: you can try to add an exponential cut-off

$$p(x) \propto x^{-\alpha} e^{-x/k}.$$

4. One can study the influence of the network structure on various processes such as *contagion* (or information diffusion, opinion formation, coordination) or *resilience*. *Assortative* networks (hubs connect to hubs, as opposed to disassortative nets, such as the web, in which the hubs are on the periphery) are more resilient. You can also look at the correlation profile: compare the number of edges between nodes of degree k and l with that in a random network; look at the average degree of the neighbours, as a function of a node's degree; look at the correlation between the degree of two adjacent nodes.

5. From a software point of view, networks can be studied with Gephi (to look at the network), NetLogo (for experiments, and to explain things to others), iGraph, NetworkX (Python), sna (R).

Cryptography I

D. Boneh (Coursera 2012)

Many historical ciphers are generalizations of the Caesar cipher (rot-3): substitution cipher, Vigenere cipher (k rot- n ciphers), Hebern machine (k substitution ciphers, forming a “rotor”), Enigma machine (idem, with 3 to 5 rotors, rotating at different speeds).

Stream ciphers use a similar idea: in the one-time pad, the message is XORed with the key, which has to be as long as the message (the safety of this cipher relies on the fact that if X and Y are binary, independent, and X is uniformly distributed, then $X \text{ XOR } Y$ is uniformly distributed); most stream ciphers use a pseudo-random generator instead, e.g., Salsa20 (or the other eStream ciphers – do not use a linear congruential generator or `random` in `glibc`).

There are many notions of a “secure” cipher. A cipher is *semantically secure* if $E(m_1, k)$ and $E(m_2, k)$ are computationally undistinguishable (for cleartexts m_1, m_2 chosen by the attacker and a random key k). A cipher is *malleable* if you can change the ciphertext to generate predictable effects on the clear text (e.g., the one-time pad is malleable: if you know that

the message starts with `From: Bob`, you can XOR it with `From: Bob XOR From: Eve` to change the sender name). There are many more, each corresponding to a type of attack or information leakage we want to prevent. Not all attacks necessitate flaws in the algorithm: it is easier to attack the implementation, e.g., with side channel attacks (measure the time, the power consumption, the cache misses, etc.) or fault attacks (overclock the device so that it outputs a wrong result – this can give some information about the key).

A **block cipher** encrypts the message by blocks, somewhat hiding its exact length. Many are built from *Feistel networks*: cut the block in two (L, R) and consider $F_1 \circ \dots \circ F_n$, where $F_i(L, R) = (R, L \text{ XOR } f_i(R))$ – the F_i are invertible (*pseudo-random permutation*, PRP) for arbitrary f_i (*pseudo-random function*, PRF). DES and AES are block ciphers. (DES and double DES are broken; there is an attack on the triple DES, but it is not practical). AES is not a Feistel network: $c_i = \sigma(f_i(c_{i-1} \text{ XOR } k_i))$ (11 times, with a permutation σ and PRP f_i , starting with $c_0 = m$).

A pseudo-random generator (PRG) $G : K \rightarrow K \times K$ defines a block cipher: it is a 1-bit PRF, and by iterating it,

$$\begin{aligned} k &\mapsto (G(k)_1, G(k)_2) \\ &\mapsto (G(G(k)_1)_1, G(G(k)_1)_2, (G(G(k)_2)_1, G(G(k)_2)_2)) \end{aligned}$$

you get a 2-bit PRF (those 4 values correspond to the encoding of 00, 01, 10, 11), and eventually an n -bit PRF (you do not need to compute all 2^n values: since you only want one branch of the tree, there are only n values to compute). The PRF can then be turned into a PRP with the Ruby-Rackoff theorem (3-round Feistel). But it is much slower than heuristic block ciphers (or the stream cipher used).

The naive use of a block cipher, cutting the message into blocks and encrypting each block separately, with the same key (ECB, Electronic Code Book), is unsafe: two identical blocks give identical cipher texts, it is not safe against replay attacks – even worse, if you encrypt an image like that, its silhouette remains visible.

Cipher Block Chain (CBC) remedies this by XORing the cleartext with the ciphertext of the previous block before encryption:

$$\begin{aligned} c_0 &= \text{IV (Initialization Vector)} \\ c_i &= E(k, c_{i-1} \text{ XOR } m_i) \end{aligned}$$

To ensure that the initialization vector is not predictable, you can use a *nonce* (“number used once”), and encrypt it (with another key):

$$\begin{aligned} c_0 &= \text{nonce} \\ \text{IV} &= E(k_1, \text{nonce}) \\ c_1 &= E(k, \text{IV}) \text{ XOR } m_1 \\ c_i &= E(k, c_{i-1}) \text{ XOR } m_i \end{aligned}$$

Randomized counter mode (CTR) uses a similar idea, but is parallelizable: XOR the message with with

$F(k, IV)$, $F(k, IV + 1)$, etc., where F is a pseudo-random function.

The **integrity** of a message can be ensured with a *message authentication code* (MAC):

$$\begin{aligned} \text{tag} &= S(\text{message}, \text{key}) \\ V(\text{message}, \text{tag}, \text{key}) &= \text{"yes"} \end{aligned}$$

Contrary to checksums, MACs have to be robust to malicious errors, not just random ones. A PRF can be used as a MAC (it should be secure and sufficiently long – more than 80 bits), for small messages. For larger messages, you can chain the MAC computations, as in, encrypted CBC-MAC:

$$\begin{aligned} c_0 &= F(\text{key}, m_0) \\ c_1 &= F(\text{key}, m_1 \text{ XOR } c_0) \\ &\dots \\ \text{MAC} &= F(\text{other key}, c_n) \end{aligned}$$

or NMAC (nexted MAC):

$$\begin{aligned} c_0 &= F(\text{key}, m_0) \\ c_1 &= F(c_0, m_1) \\ &\dots \\ \text{MAC} &= F(\text{other key}, c_n \parallel \text{pad}) \end{aligned}$$

(without the last encryption, it is not secure: an attacker can append something to the message).

Padding the last block with zeroes is unsafe. Instead, you can add $100 \dots 00$, but you may need to add a dummy block. CMAC provides padding with no dummy block: replace the final encryption with

$$\begin{aligned} c_n &= F(\text{key}, (m_n \parallel 100 \dots 00) \text{ XOR } \text{key}_1) \\ c_n &= F(\text{key}, m_n \text{ XOR } \text{key}_2) \end{aligned}$$

depending on whether there is padding or not.

The MAC can be parallelized (PMAC):

$$\begin{aligned} c_1 &= F(\text{key}_1, m_1 \text{ XOR } P(\text{key}_0, 1)) \\ c_2 &= F(\text{key}_1, m_2 \text{ XOR } P(\text{key}_0, 2)) \\ \text{return } &F(\text{key}_1, c_1 \text{ XOR } \dots \text{ XOR } c_n) \end{aligned}$$

If F is invertible (a PRP rather than just a PRF), you can invert the last step, add some more data, or modify one of the blocks, and recompute the tag.

A one-time MAC can be computed from a large prime number q (e.g., $2^{128} + 51$), using two random numbers k , $a \in \llbracket 1, q \rrbracket$ as a key: if the message (m_1, m_2, \dots) is made of 128-bit integers, the tag is $\sum_i m_i k^i + a \pmod q$.

A one-time MAC can be transformed into a many-time MAC (Carter-Wegman MAC):

$$\text{tag} = (r, F(\text{key}_1, r) \text{ XOR } S(\text{key}_2, \text{message}))$$

where r is random, S is a 1-time MAC, F is a PRF. It is a random MAC: it is not deterministic.

From a collision-resistant compression function operating on small messages, you can build a collision-resistant function that operates on larger messages, by chaining the hash function: e.g., SHA-1 (almost broken), SHA-256, SHA-512, Whirlpool (slower) .

A collision-resistant function is not a secure MAC: use a Hash MAC (HMAC) instead (it is very similar to NMAC, which used two keys, and a PRF instead of a hash):

$$S(\text{key}, \text{message}) = \text{Hash}(\text{key} \parallel \text{outer pad}, \text{Hash}(\text{key} \parallel \text{inner pad}, \text{message})).$$

When checking if the MAC is correct, you need to compare several bytes: the compiler (lazy language, optimizing compiler, `||/∆∆` shortcut, etc.) may want to stop as soon as it knows the result, telling a potential attacker whether the first byte was incorrect. Before the comparison, you can encrypt the correct MAC and the MAC to test, and compare the encrypted values.

Encryption alone is not secure against tampering (it is safe against passive attacks only, not active attacks): always use **authenticated encryption** instead. We want a new notion of safety, *ciphertext integrity*: the recipient can reject messages if they are not valid; the attacker cannot create a valid ciphertext; the attacker chooses the ciphertext. This does not provide any safety against replay attacks

Encrypt-and-MAC (add the MAC of the plaintext and send it in clear) is not safe (it leaks information); MAC-then-encrypt (add the MAC of the plaintext and encrypt everything) is only safe in some special cases; encrypt-then MAC (add the MAC of the ciphertext, in clear) is safe.

There are many standards that combine a cipher and a MAC: TLS (to prevent replay attacks, the sender and the recipient keep track of a counter, incremented each time, used in the MAC computation, but never exchanged; suffers from padding attacks, timing attacks, information leakage (the reason why a message is rejected); renegotiates the key in case of a problem to avoid those attacks); 802.11b (MAC-then-encrypt (the reverse is safer), linear CRC (not a cryptographic MAC), repeated IV, related keys, etc.); SSH (the length of the message is not authenticated).

In many situations, you have one key, but the algorithm needs several: you can use a *key derivation function* (KDF). You could use a PRF, if the source key is uniform. Use a salt (non-secret, fixed, chosen at random) to extract a pseudo-random key from the source key, $K = \text{HMAC}(\text{salt}, \text{source key})$ and use HMAC as a PRF with key K . Do not use password-based KDFs: there is not enough entropy, and dictionary attacks work well – if you insist on using them, use a salt and a slow hash function, e.g., iterate $x \mapsto \text{hash}(x \parallel \text{salt})$ many (10^6) times.

Deterministic encryption (no nonce, the ciphertext is always the same) allows you to search in an encrypted database but is unsafe, if the messages are small or repeated (e.g., foreign keys).

Tweakable encryption, $E(\text{key}, \text{tweak}, \text{message})$, where each $E(k, t, \cdot)$ is a PRP, is often used in disk encryption (e.g., xts), with t the sector number and $E(k, t, x) = E(E(k, t), x)$.

Key exchange can be performed through a trusted third party (TTP), who shares a key with everyone: the TTP chooses a random key K_{AB} , encrypts it for A as $E(K_A, K_{AB})$, sends it to A , encrypts it for B as $E(K_B, K_{AB})$, and sends it to A (not B , A will forward to B). This is safe against eavesdropping, but not against active attacks or TTP corruption.

In the *Merkle puzzle* scheme, A sends n puzzles to B , each taking $O(n)$ time to solve. B chooses one, solves it, and the solution is of the form (id, key) (A had generated n different keys). B sends the id to A ; they both know the key. There is a gap between the work done by B and the attacker: $O(n^2)$; it is very inefficient.

Public-key encryption relies heavily on number theory:

- Euclid's theorem (there exists u and v so that $au + bv = \gcd(a, b)$) can be used to compute inverses in $\mathbf{Z}/n\mathbf{Z}$.
- Fermat's theorem ($x^{p-1} = 1 \pmod{p}$) can also be used to compute inverses ($x^{-1} = x^{p-2}$), but is less efficient ($(\log p)^3$ instead of $(\log p)^2$). It can also be used to select random prime numbers: pick p at random, until $2^{p-1} = 1 \pmod{p}$; You can be fairly confident ($P[\text{not prime}] < 2^{-60}$), but there are better, non-probabilistic, tests of primality.
- Euler: \mathbf{F}_p^\times is a cyclic group.
- Lagrange: $\text{ord}_p(g) | p-1$ (the order of an element of an abelian group divides the order of the group) can be used to prove Fermat's theorem.
- Fermat: $x^{p-1} = 1 \pmod{p}$.
- Fermat: $x^{\phi(n)} = 1 \pmod{n}$ if $x \in (\mathbf{Z}/n\mathbf{Z})^\times$ (a special case of Lagrange), where $\phi(n) = |(\mathbf{Z}/n\mathbf{Z})^\times|$ is Euler's totient function.

Many algorithms rely on the difficulty of the following problems:

- Computing a modular e -th root: the best known algorithms require a factorization of the modulus. In \mathbf{F}_p , $x > 0$ is a quadratic residue (i.e., a square) iff $x^{(p-1)/2} = 1 \pmod{p}$ (Legendre symbol); if $p = 3 \pmod{4}$ and c is a quadratic residue, then $\sqrt{c} = c^{(p+1)/4}$; if $p = 1 \pmod{4}$, the square root can be found in $O((\log g)^3)$, with a randomized algorithm; you can also solve quadratic equations with the high-school formula.
- Inverse in $\mathbf{Z}/n\mathbf{Z}$;
- Roots of a polynomial of arbitrary degree in \mathbf{F}_p ;
- Discrete logarithm mod p : find x so that $g^x = a \pmod{p}$ (believed to be hard for \mathbf{F}_p^\times (p large) and elliptic curves; can be computed with the GNFS algorithm for \mathbf{F}_p^\times); can be used to create a collision-

resistant hash: $H(x, y) = g^x h^y$ in $G = \mathbf{F}_p^\times$, with $g, h \in G$ (key) – finding a collision is equivalent to computing a discrete log;

- Factor a number into a product of two large primes (soon possible for 1024-bit numbers, hard for 2048-bit numbers)
- Find a root of a polynomial of degree $> 1 \pmod{n}$ (the best known algorithms factor n).

In the *Diffie-Hellman* scheme, choose a large prime p , and g a generator of the cyclic group $(\mathbf{Z}/p)^\times$. A chooses a in $\{1, \dots, p-1\}$, sends $g^a \pmod{p}$ to B ; B chooses b in $\{1, \dots, p-1\}$, sends $g^b \pmod{p}$ to A . They both compute g^{ab} and use it as a key. For a 256-bit key, you need a 16,000-bit prime. The same idea works with other cyclic groups, e.g., elliptic curves: a 512-bit curve suffices. It is insecure against man-in-the-middle attacks. It can work passively: everyone publishes g^a , and can read the g^b of everyone else. It only works to communicate between two parties; it can be generalized to 3 but, beyond that, the problem is still open.

Many public-key encryption schemes are based on trapdoor encryption: a pair of functions that can be generated randomly, one inverse of the other, but otherwise very difficult to invert. One is non-deterministic and used to encrypt messages (public key), the other is deterministic (private key, trapdoor).

The RSA functions are $x \mapsto x^d$ and x^e , where $N = pq$ is a product of two large prime numbers, $\phi(N) = (p-1)(q-1)$, x invertible mod N (most are), d is random and e is such that $de = 1 \pmod{\phi(N)}$; the functions are inverses because $x^{\phi(N)} = 1 \pmod{N}$. It can be used to exchange the key for a symmetric cipher (rather than encrypt messages, as some textbooks suggest: the functions are deterministic).

We do not know if RSA is safe: we do not know if computing e -th roots is as hard as factoring. Encryption is much faster than decryption (other public key encryption algorithms are more balanced: ElGamal). There are timing or power attacks: computing $c^d \pmod{N}$ can expose d ; there are fault attacks: a single error in the computation of $c^d \pmod{N}$ (check, by elevating to the e -th power: it should be 1) can leak the decomposition of N . There are key generation problem: p is generated before q , when there is not enough entropy: many routers/web servers have the same p (and different q): compute $\gcd(n_i, n_j)$ for all the pairs of routers/web servers you can find – if it is not 1, you have factored n_i (0.5% of web servers are affected).

PySP: Modeling and solving stochastic programs in Python
J.P. Watson et al. (2011)

Stochastic programs can be converted into deterministic programs (the objective function, an expectation, becomes a weighted sum; each decision variable is duplicated for each scenario; the constraints are likewise duplicated; you just have to add non-anticipatory constraints, that encode the structure of the scenario tree),

but the size of the resulting problem (most optimizers have a pre-solver, that identifies and removes redundant variables and constraints, but this only halves the size of the problem) makes it difficult to solve, unless you exploit the structure of the problem.

Vertical strategies, such as *Bender's decomposition* (*L*-shaped method, for 2-stage problems) or nested decomposition schemes decompose the problem by stages. Horizontal strategies, such as **progressive hedging** or the dual decomposition, decompose it by scenarios.

Progressive hedging suggests to:

- Solve the deterministic problems for all the scenarios;
- Average the solutions (the result does not satisfy the non-anticipatory constraints);
- To each sub-problem, add a penalty for the breached non-anticipatory constraints, re-solve the problems, and iterate.

The algorithm can be improved:

- The first iterations do not require a precise solution;
- Detect if a variable converges early; if so, fix its value;
- Detect cycles (tabu search, with a Bloom filter), which are common with integer variables;
- The sub-problems can be solved in parallel.

Pyomo: modeling and solving mathematical programs in Python
W.E. Hart et al. (2011)

The article stresses the need for more modularity in mathematical programming software: one should separate the structure of the problems (abstract model) from the data (concrete model) and the solver; use a high-level programming language (Python, as opposed to AMPL, AIMMS, GAMS), which makes it easy to enumerate sets of constraints and allows you to implement algorithms that require many optimizations (e.g., branch-and-cut; Bender's decomposition for stochastic programs; problems with an exponentially large number of constraints in which you can quickly identify the breached constraints, add them to the problem and iterate).

Coopr contains: Pyomo, to specify the problem; and interface to different solvers; a parallelization framework, build from pickle (serialization) and pyro (RPC), should you want to solve several subproblems in parallel.

The (only?) open-source solvers supported are CBC, GLPK, IPOPT, from the COIN-OR project.

Alternatives include:

- FlopC++: a problem is a C++ class, derived from `MP_Model`; thanks to operator overloading, the syntax looks readable – but I suspect the error messages are not;
- OptimJ, a Java extension – not a library, but an extension to the language, with its compiler, integrated with Eclipse, no longer commercially developed.

The Cornish-Fisher expansion in the context of delta-gamma-normal approximations
S.R. Jäschke (2001)

Proof of the Cornish-Fisher expansion: a series expansion of $F^{-1} \circ \Phi$, with F the cumulative distribution function (cdf) of interest and Φ a known cdf, typically Gaussian), often used to estimate the value at risk (VaR) of non-gaussian distributions. The error increases as you progress into the tail, and the approximated VaR need not even be monotonic. However, the errors from the Cornish-Fisher expansion are smaller than those coming from a quadratic approximation (Δ - Γ -normal): to be safe, compare with a Monte Carlo estimator. Using a reference distribution Φ with fatter tails may reduce those problems.

Asset liability management for individual households
M.A.H. Dempster and E.A. Medova (2010)

There are three ways of solving ALM problems: dynamic programming; stochastic programming (transform the stochastic program into a huge, sparse, deterministic, linear program); various heuristics. Dynamic programming suffers from the curse of dimension; so does stochastic programming, but to a lesser extent. The authors model 10 asset classes (monthly, but the investment and consumption decisions are annual) with geometric brownian motions (equities, bonds, commodities, alternative investments, real estate) or geometric Ornstein-Uhlenbeck processes (cash, inflation, treasury bill rate), with correlated noises, from market data (for some asset classes, you have to compute the total returns from the prices, by adding dividends or coupon payments), and use the corresponding scenarios in an ALM problem, looking for the optimal consumption and investment decisions, to maximize the expected utility of life-time consumption. The utility is piecewise linear and specified separately for each goal; it is penalized for bankruptcy (excess borrowing). The simulation also includes the death of the investor (variable investment horizon), and fixed, inflation-indexed or growing liabilities. Consumption is divided between equity-preserving goals (e.g., housing), often requiring a downpayment and subsequent mortgage payments (making the problem path-dependent), and non-capital goals.

Individual asset liability management
E.A. Medova et al. (2008)

Presentation of the corresponding GUI, to specify the ALM problem and analyze the results.

Discretionary Wealth Hypothesis and ALM
D. diBartolomeo (2010)

Tackle the dynamic asset allocation problem with transaction costs as follows: always invest in a mean-variance efficient portfolio, but change the risk aversion parameter according to the following rule of thumb,

which is not unlike portfolio insurance (PI),

$$\lambda = \frac{1}{2} \frac{\text{Assets}}{\text{Assets} - \text{PV}(\text{Liabilities})}$$

and add a penalty to the mean-variance optimization to account for the transaction costs

$$\frac{\text{transaction costs} \times \text{turnover}}{\text{probability that the new portfolio is better}}.$$

The present value of the liabilities is computed using a risk-free recombining binary tree of short interest rates, so that $\text{Assets} - \text{PV}(\text{Liabilities})$ is non-negative iff the probability of bankruptcy is zero.

The 401(k) retirement income risk
F. Sortino and D. hand (2011)

Since the goal of retirement investment is not to beat the market, but to “retire with dignity”, returns and volatility are not the right benchmark. Instead, to decide if you should make a catch-up contribution, you can look at:

- The probability of achieving the goal;
- The present value of the assets and liabilities;
- The *desired target return* (DTR), *i.e.*, the internal rate of return required for this present value to become positive.

Optimal investment strategies in defined contribution pension plans
D. Blake et al. (2011)

Most pension finds separate the accumulation and decumulation phases, asking the wrong questions (how much do you want to save?), using incorrect targets (performance, rather than retirement), failing at their market timing attempts (when volatility is high, returns are low: one should reduce the weight of equities) and leading to under-funded pension plans.

Only in rare cases (power utility and iid asset returns) is the 1-period optimal strategy also a multi-period optimal strategy. The Kelly principle can be generalized to account for (risky) labour income; the resulting strategy, “stochastic lifestyle”, is reminiscent of the discretionary wealth hypothesis.

Deciding if/when to annualize is an option exercise problem (it depends on age, wealth and risk aversion). Since most pensions are underfunded, most people will buy an annuity; to avoid low interest rates, it is preferable to spread its purchase over time (phased annuity purchases).

It is not advisable to seek an absolute guarantee to deliver the desired pension: such guarantees are very expensive to secure.

Duration-enhancing overlay strategies for defined-benefit pension plans
J.M. Mulvey et al. (2011)

Defined-benefit pension plans have fixed liabilities: they are bonds, with a very high duration (10 to 15 years). Since long-duration assets (long-term bonds) have lower returns than short-duration assets (stocks, etc.), you may want to invest in a core portfolio of short-duration assets (100%), and a *duration-enhancing overlay* (50% to 100%, long long-duration bonds, short short-duration bonds) to hedge the interest rate risk.

A robust optimization approach to pension fund management
G. Iyengar and A. Ka Chun Ma (2011)

The defined-benefit pension problem can be formulated as follows: find the future contributions w_t and the portfolio composition \mathbf{x}_t (equity index and bonds of each maturity) to minimize the present value of the future contributions, subject to the requirement to meet the prescribed (deterministic) liabilities at all times t , and regulatory requirements

$$\text{NAV} \geq \beta \text{PV}(\text{future liabilities}).$$

The time horizon T is sufficiently long so that liabilities beyond T are negligible. (The problem is slightly more complicated: since the liabilities have to be met, the firm has to find the money somewhere – it is a corporate finance decision, involving the use of debt vs equity vs retained earnings, taxes (interest payments are tax-deductible), debt rating, etc.).

The constraints are often replaced by **chance constraints**, asking that the probability that the constraint is breached be below some threshold ε .

Equity prices follow a geometric brownian motion; bond yields are described by the Nelson-Siegel model, whose factors follow a mean-reverting process; the innovations for those four factors are correlated.

The chance constraints $P[\text{constraint breached}] \leq \varepsilon$ are of the form $P[X_t \geq a] \leq \varepsilon$, where X_t is a stochastic process defined from the risk factors. Using Ito’s formula, you can compute its stochastic differential equation, and then linearize it at $t = 0$, *i.e.*, assume it is just a brownian motion. The constraint becomes

$$P[\text{something gaussian}] \leq \varepsilon;$$

it can be written as a second-order cone (SOC) constraint $\|\mathbf{B}\mathbf{x} - \mathbf{a}\| \leq \mathbf{d}'\mathbf{x} + c$.

To ensure that all K chance constraints are satisfied with probability $1 - \varepsilon$, use Bonferroni’s inequality: assume they are independent and set their thresholds to ε/K . The resulting (SOC) optimization problem is no longer stochastic: the contributions w_t and numbers of shares \mathbf{x}_t do not depend on the state of the world at time t .

The resulting strategy is conservative, perhaps because of the linearization of the geometric brownian motion.

**Alternative decision models
for liability-driven investment
K. Schwaiger et al. (2011)**

There are many ways of formulating the pension problem, each simplifying a different aspect of it. Here are four of them.

- A deterministic program, in which you minimize both the sum of all the contributions and the sum of the absolute values of the PV01 of the assets and liabilities (since there are two objectives, the result is not a single strategy, but an efficient frontier of strategies).
- A stochastic model (you have to generate scenarios), with two stages (to simplify), in which you match the present value (PV, not PV01) of the assets and liabilities; the two objective functions are the expected contributions and the expected absolute value of the difference in present values.
- A chance constraint can be added to this model

$$P[\text{Assets} - \gamma \text{Liabilities} > 0] \geq 1 - \varepsilon.$$

The equivalent deterministic formulation (the constraint is breached in at most εN of the N scenarios) involves binary variables.

- The chance constraints can be replaced by **integrated chance constraints** (this is similar to the difference between value at risk and expected shortfall)

$$E[\text{Assets}_t - \gamma \text{Liabilities}_t] \leq \text{PV}_t(\text{Liabilities}(\omega));$$

binary variables are no longer required.

**A liability-relative drawdown approach
to pension asset liability management
A. Berkelaar and R. Kouwenberg (2011)**

Find the portfolio weights \mathbf{w} (constant over time) to maximize the (logarithmic) utility of the funding ratio Assets/Liabilities, whose final value can be approximated as

$$\prod_t (1 + \mathbf{w}' \mathbf{r}_t - r_t^{\text{liabilities}}) \approx \exp \sum_t (\mathbf{w}' \mathbf{r}_t - r_t^{\text{liabilities}}),$$

with a constraint on the variance of the surplus (Assets – Liabilities) or on the maximum funding ratio drawdown (linear program) or on the **conditional** funding ratio **drawdown** (average of the worst 10% worst drawdowns, also a linear program)

Allow for some short-selling: many pension-funds use a return-generating portfolio and a liability-hedging overlay.

The scenarios (returns for equities, commodities, real estate, hedge funds) are generated from risk factors (“state variables”: level, slope and curvature of the yield curve, default spread (Baa – Aaa), consumption-wealth ratio) through a vector autoregressive (VAR) model:

- The yield curve is described by the **Diebold-Li model**, *i.e.*, the Nelson-Siegel model in which the parameters (level, slope, curvature) form a VAR process;
- Use a 2-step model to deal with time series with a shorter history (hedge funds, etc.): first model (VAR) the long-history variables, then model the short-history ones, adding the long-history ones as exogenous variables;
- Modify the intercept of the VAR model to avoid unrealistic reversion to the (old) long-term averages.

The term structures of return volatility, for different asset classes, are very different.

**Dynamic risk management:
optimal investment with risk constraints
S. Jarvis (2011)**

The dynamic asset allocation problem can be solved with partial differential equations (PDEs).

Asymmetric risk measures are even more important for dynamic strategies than for static ones: they can create very non-gaussian return distributions – prefer the conditional value at risk (CVaR, expected shortfall) to the standard deviation.

The optimal dynamic strategy that maximizes the expected CRRA (constant relative risk aversion) utility $((x^{1-\gamma} - 1)/(1 - \gamma))$ or $\log x$ if $\gamma = 1$) of final wealth, when asset prices follow a geometric brownian motion. is the *growth-optimal portfolio* (GOP, a generalization of the Kelly principle); with a CVaR constraint, the payoff is transformed and looks like a collar \nparallel . (This is like an asset and liability management (ALM) problem, but with no liabilities.)

The optimal strategy can be computed by dynamic programming on a scenario tree or, better, a recombining tree. Alternatively, since the probability distribution of the payoff is solution of the **Fokker-Planck equation** (Kolmogorov forward equation – not to be confused with the Kolmogorov backwards equation, which gives the option price), one can solve it numerically, for many investment strategies, and keep the one with the highest utility. (Contrary to option pricing, the payoff is unknown: we are looking for the strategy that maximizes the utility of the payoff.) The resulting distribution looks like a soft CPPI \mathbb{L} (constant proportion portfolio insurance).

While the CVaR is an acceptable risk measure for static (1-period) strategies, it is less so for dynamic ones. A measure of risk ρ can often be seen as a *capital requirement*: $\rho(X)$ is the amount of capital needed to make X acceptable, *i.e.*, $\rho(X) = \text{Min}\{m : \rho(X + m) \leq 0\}$; therefore $\rho(X + m) = \rho(X) - m$. In a multi-period setting, one would expect $\rho_t(X_T) = \rho_t(-\rho_{t+1}(X_T))$ – but this does not hold for the CVaR. The only **time-consistent risk measures** are the *entropic risk measures*

$$\rho_t(X) = \frac{1}{\gamma} \log E_t[\exp -\gamma X]$$

or, more generally (if you remove the translation invariance $\rho(X + m) = \rho(X) - m$), certainty-equivalent risk measures

$$\rho_T(X) = u^{-1} E_t[u(X)].$$

[The consequence of time-inconsistency, the fact that one is risk-seeking when the wealth is very high or very low, looks harmless to me.]

Bank asset-liability and liquidity risk management
M. Choudhry (2010)

Asset liability management (ALM), in a bank, focuses on:

- *Liquidity risk*: the dates of the assets and liabilities do not match (they are even sometimes unknown: options, current accounts, credit lines, etc.), so the bank will need to refinance some of them, and there is no guarantee that this will be possible at a reasonable price;
- *Interest rate risk*: even if the dates match, the rates need not match.

A two-factor HJM interest rate model for use in asset liability management
S. Kaya et al. (2010)

The G_{2++} short rate model (a 2-factor Gaussian model) can be generalized to a term-structure (HJM) model, and used to price structured products.

Asset liability management modelling with risk control by stochastic dominance
X. Yang et al. (2010)

Stochastic dominance (first order, second order, and their variants: interval second order, etc.) constraints can be included in optimization or stochastic optimization problems (they become *chance constraints*).

Zero-coupon yield curve estimation with the package termstrc
R. Ferstl and J. Hayden (JSS 2010)

The yield curve is often modelled empirically, from market data, either in a non-parametric way (linear interpolation of discount factors (bad) or of their logarithms (better), piecewise interpolation of the instantaneous forward rate (equivalent), splines) or by assuming that the curve has a specific shape (Nelson-Siegel, Svensson – beware, the formula is different if you model the yield or the instantaneous forward rate), sometimes with a (VAR, VARMA) time-dependence on the parameters (Diebold-Li). Here are some implementations caveats:

- To avoid identifiability problems, you may want to impose a few constraints, e.g., $\tau_2 - \tau_1 > \varepsilon$ in the Svensson model;

- Since the problem is non-convex, you may want to use a grid search (on the non-linear parameters: τ_1 , τ_2) to find good starting values;
- The problem is heteroskedastic: use weights (e.g., the inverse of the duration, or the bid-ask spread).

The implementation, in the `termstrc` package, is unusable: prefer `fBonds` (it uses a grid search, but no constraints, no weights, no time-dependence, and models forward rates rather than yields).

Foundations of the Statpro simulation model
M. Marchioro and D. Cintioli (2007)

To estimate risk (value at risk, expected shortfall), resample from the multiplicative (stock prices, implied volatility, and other positive quantities) or additive (interest rates, swaps, and other quantities that could become negative) historical changes.

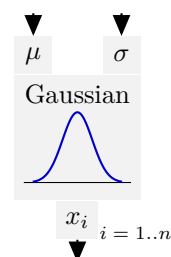
For fixed income, resample the instantaneous forward rates rather than the prices. For complex instruments, price them from simple instruments. For large portfolios, pay attention to missing data (coming, e.g., from different holidays).

Pricing simple interest-rate derivatives
M. Marchioro (2008)

Clear presentation of the annoying accounting conventions (day counting convention, business day convention, compounding convention) and overview of the simplest common interest rates derivatives (forward rate agreements (FRA) and swaps). The interest rate term structure (“yield curve”) can be estimated from deposit rates (the -ibor rates: overnight, tomorrow-next, $t+2, \dots$, until 1 year) and the swap rates (beyond 1 year: since the principal is not exchanged, the risk is lower). To interpolate the term structure, linear interpolation of the logarithm of the discount factor, *i.e.*, piecewise constant interpolation of the forward rates, gives decent results.

Doing bayesian analysis, a tutorial with R and BUGS
J.K. Kruschke (2011)

Leisurely introduction to Markov Chain Monte Carlo (MCMC) computations in bayesian statistics, in which the graphical representation of bayesian models as a combination of building blocks is more pictorial than the traditional BUGS one.



Introduction to complex systems (Markov models, Lyapunov functions, random walks, Polya's urn model, chaos, network models), agent-based models (Schelling's segregation model, epidemics models (SIR, SIS), coordination game, replicator dynamics, wisdom/madness of crowds), game theory (prisoner's dilemma, auctions, colonel Blotto) with applications in economy (Solow growth model and the role of innovation), sociology, politics (voting and aggregation of preferences), etc., with examples in NetLogo – useful if you need to explain an agent-based model to a non quantitative audience.

Algorithms, design and analysis I
T. Roughgarden (Coursera 2012)

Introductory algorithms course, covering:

- Sorting: insertion, selection, bubble-sort, merge-sort, quick-sort; bucket-sort (if the data is $U(0,1)$), counting sort (discrete, repeated values), radix sort;
- Divide-and-conquer: multiplication of large integers or matrices (with the Karatsuba and Strassen improvements); number of inversions in an array; closest pair; median (adapt quick-sort; or split the data into groups of 5; recursively compute the median of medians; use it as a pivot)
- Heuristics for NP-complete graph problems: graph colouring to assign temporary variables to registers in a compiler (pick a node with at most k neighbours, remove it, iterate, backtrack if needed, remove a node if stuck), minimum cut (select an edge at random, contract it, iterate, repeat and keep the best solution);
- Graph search: breadth-first search (BFS: use a queue, FIFO), to compute distances or connected components; depth-first search (DFS, backtracking: use a stack or recursion), to compute a topological ordering (all the edges go forward) of a directed acyclic graph (you could also look for a sink vertex, remove it, and iterate) or its strongly connected components (DFS on the reverse graph, remember the finishing time of each vertex, DFS in the reverse order)
- Heaps (or priority queues): minimum paths from a vertex, running median (use two heaps); balanced binary search trees; union-find (aka disjoint-set) to store partitions, with two operations, union and find (the component an element is in, e.g., via a representative element), as a forest of trees (components), each child pointing to its parent, with short paths to the root (representative element) – for more efficiency, when running find(), have all the elements you see point to the root.
- Hash tables (if you implement your own, use a random hash function); Bloom filter (use an array of n bits and k hash functions; to insert an element, set the k corresponding bits to 1);

**Co-movement of energy commodities revisited:
evidence from wavelet coherence analysis**
L. Vacha and J. Barunik (2012)

Wavelets can be used to analyze the coevolution of two time series: let

$$W_x(u, s) = \frac{1}{\sqrt{s}} \int_{-\infty}^{\infty} x(t) \bar{\psi}\left(\frac{t-u}{s}\right) dt$$

be the continuous wavelet transform of a time series x (at time t and scale s); the squared **wavelet coherence** coefficient is

$$R_{x,y}^2(u, s) = \frac{|S[s^{-1}W_x(u, s)\bar{W}_y(u, s)]|^2}{S[s^{-1}|W_x(u, s)|^2]S[s^{-1}|W_y(u, s)|^2]}$$

where S is some smoothing operator. It measures the linear correlation at a given location and scale. The wavelet coefficient phase difference

$$\phi_{x,y}(u, s) = \arg S[s^{-1}W_x(u, s)\bar{W}_y(u, s)]$$

can highlight delays in the oscillations of the two time series (add it as arrows (as a vector field) on top of the wavelet coherence heatmap plot).

**On Hurst exponent estimation
under heavy-tailed distributions**
J. Barunik and L. Kristoufek (2012)

Empirical comparison of various Hurst exponent estimators: rescaled range (R/S), multifractal detrended fluctuation analysis (MF-DFA), detrending moving average (DMA), generalized Hurst exponent (GHE, *i.e.*, scaling of the q -moments, often with $q = 2$) – GHE has lower variance and bias.

Monte-Carlo-based tail exponent estimator
J. Barunik and L. Vacha (2012)

The Hill estimator of the tail exponent is biased on small samples – use a simulation-based estimator instead:

- Generate random samples, with the same size, from various stable distributions;
- Compute their Hill estimators, for various threshold choices;
- Compare with the Hill estimators (for all those thresholds) of the actual data (they use the L^1 norm and give the same weight to all the thresholds).

Properties of the most diversified portfolio
Y. Chouefaty et al. (2011)

A few remarks on the **diversification ratio** (ratio of the portfolio's weighted volatility to its overall volatility), a generalization of the Herfindahl index (entropy) when assets are correlated and/or do not have the same risk. that can be interpreted as the (square of the) effective number of (uncorrelated) assets (or risk factors) in the portfolio. (This approach is limited to unleveraged, long-only portfolios.)

***Orthogonalized equity risk premia
and systematic risk decomposition***
R.F. Klein and K.V. Chow (2010)

In a factor risk model, you may want the factors to be orthogonal, e.g., to have a simple risk decomposition. Gram-Schmidt orthogonalization depends on the order of the factors, but there is a symmetric variant: let $F = (f_1 | \dots | f_n)$ be the matrix of (centered) factor returns; we want S so that $\text{Var } FS = I$, i.e., $S'F'FS = I$, i.e., $SS' = (F'F)^{-1}$, i.e., $S = (F'F)^{-1/2}C$ for an arbitrary $C \in O(n)$, e.g., $C = I$.

***Derivatives and credit contagion
in interconnected networks***
S. Heise and R. Kühn (2012)

Most models of contagion in financial networks (bipartite, firms vs banks and insurance companies) only take into account who lends to whom, and neglect synthetic credit risk exposure through CDSes – it is not negligible, and taking it into account requires adding hyperedges to the network (between three nodes: firm-bank-insurance or firm bank-bank).

***Second-order price dynamics:
approach to equilibrium with perpetual
arbitrage***
E. Kempt-Benedict (2012)

The presence of arbitrageurs and rationally-bounded economic actors explains the permanent fluctuations around the equilibrium. (This is the usual informed traders vs noise traders game, expressed in terms of supply and demand.)

***A multifractal approach
towards inference in finance***
O. Løvstetten and M. Rypdal (2012)

Multifractal processes have stationary increments and satisfy $E[|X_t|^q] \propto t^{\zeta(q)}$ for some concave function ζ (linear for self-similar (i.e., stable) processes such as Brownian motion or Lévy flight). There are many ways of building such processes or their discrete equivalents. For instance (multi-fractal random walk):

$$\begin{aligned}x_n &= X_n - X_{n-1} \\x_n &\sim N(0, \sigma_n) \\ \log \sigma_n &\sim \text{Gaussian Process} \\ \text{Cov}(\log \sigma_t, \log \sigma_s) &\propto \log^+ \frac{T/\Delta t}{|t-s|+1}\end{aligned}$$

***Numerical methods and optimization
in finance***
M. Gilli et al. (2011)

The first part of the book covers many numerical topics, such as the inherent imprecision of floating-point arithmetic (for which $\sum 1/k$ converges), numerical instability (the algorithm amplifies rounding errors), ill-conditioning (small changes in the input lead

to large changes in the output), the conditioning number ($|xf'(x)/f(x)|$, or $\kappa(A) = \|A\| \|A^{-1}\|$ for linear functions), matrix decompositions (LU, QR, SVD, Cholesky), **structural rank** (the maximum rank of a sparse matrix with the same presence/absence pattern), finite differences (implicit, explicit and θ methods – Crank-Nicolson is obtained for $\theta = 1/2$), flaws in pseudo-random number generators and the way they are used (resetting the seed before each number may not be a good idea), low-discrepancy sequences, bootstrap (the **Taylor-Thompson algorithm** is a smooth non-parametric bootstrap: take a point at random, look at its k nearest neighbours, move it towards them by a random amount, iterate). Examples include European and American option pricing, time series models (with a clear interpretation of the model parameters), CPPI, VaR estimation.

The second part is devoted to (unconstrained) optimization. The **fixed point method** (rewrite the problem as $x = f(x)$ and iterate $x_{n+1} = f(x_n)$ until convergence) is more useful than one may think: it is the base of the Jacobi, Gauss-Seidel and SOR algorithms, used to solve large linear systems (e.g., those coming from PDEs) or large non-linear systems; it can also be used to compute the *S-estimator* (a robust estimator whose cost function is defined implicitly). Gradient-based algorithms (gradient descent, quasi-Newton, Levenberg-Macquardt), heuristics (local search, tabu search, simulated annealing, threshold acceptance, genetic algorithms, differential evolution, particle swarm optimization, ant colony optimization) and hybrid methods are also presented, with examples from portfolio optimization, term structure models, option pricing model calibration (via the characteristic function), robust regression (least median of squares, least quantile of squares, least trimmed squares).

The book contains many plots, but the lack of titles and labels makes them difficult to read.

A tutorial on geometric programming
S. Boyd et al. (2007)

A *monomial* is a term of the form $cx_1^{a_1} \dots x_n^{a_n}$, with $c > 0$ – but the a_i are arbitrary real numbers. A **posynomial** is a sum of monomials. A **geometric program** (GP) is an optimization problem of the form “minimize P_0 , so that $P_1, \dots, P_n \leq 1$ and $M_1 = \dots = M_m = 1$ ”, where the P_i are posynomials and the M_j monomials. The exponents can be arbitrary, but the coefficients have to be positive and the equality constraints only have one term.

Geometric programs can be solved efficiently via interior point methods: if f is a (generalized) posynomial, then $\log f(e^y)$ is convex (in terms of f , the convexity condition involves geometric averages, hence the name).

Surprisingly many problems are geometric programs, or can be reformulated as such, or can be approximated by GPs. Here are some of the transformations (defining

generalized posynomials).

$$\begin{aligned} f(x) - m(x) \leq 0 &\rightarrow f(x)/m(x) \leq 1 \\ f(x)^{2.2} + g(x)^{3.1} \leq 1 &\rightarrow f(x) \leq t_1, g(x) \leq t_1, \\ &\quad t_1^{2.2} + t_1^{3.1} \leq 1 \\ f_0(f_1(x), \dots, f_n(x)) \leq 1 &\rightarrow f_0(t_1, \dots, t_n) \leq 1, \\ &\quad f_1(x) \leq t_1, \dots, f_n(x) \leq t_n \\ \text{Max}\{f_1(x), f_2(x)\} \rightarrow t, &\quad f_1(x) \leq t, f_2(x) \leq t \end{aligned}$$

Trade-off analysis measures the impact of the constraints: replace the constraints $f_i(x) \leq 1$ and $g_j(x) = 1$ with $f_i(x) \leq u_i$ and $g_j(x) = v_j$, consider the value $p(u, v)$ of the corresponding problem, and compute the partial derivatives (the logarithm disappears because we are evaluating them in 1)

$$\begin{aligned} S_i &= \left. \frac{\partial \log p}{\partial \log u_i} \right|_{u=v=1} = \left. \frac{\partial \log p}{\partial u_i} \right|_{u=v=1} \\ T_j &= \left. \frac{\partial \log p}{\partial \log v_j} \right|_{u=v=1} = \left. \frac{\partial \log p}{\partial v_j} \right|_{u=v=1}. \end{aligned}$$

You do not have to explicitly compute those quantities: they come from the dual problem. This is also useful for infeasible problems: the sensitivities (using some measure of infeasibility as objective, or a penalized objective) tell you which constraints bear the most responsibilities.

Here are more transformations

$$\begin{aligned} \frac{p(x)}{r(x) - q(x)} + f(x) \leq 1 &\rightarrow t + f(x) \leq 1, \\ q(x) + p(x)/t &\leq r(x) \end{aligned}$$

and approximate transformations (functions whose Taylor expansions have positive coefficients – for the exponential and the logarithm, the problem can be solved exactly: after the logarithmic transformation, the function is convex).

$$\begin{aligned} \exp f(x) &\underset{f(x) \approx b}{\approx} e^b \left(1 + \frac{f(x) - b}{a} \right)^a \\ \log q(x) &\underset{a \gg 1}{\approx} a q(x)^{1/a} - a \end{aligned}$$

There are many generalizations:

- in a mixed linear GP, some variables appear as GP terms, others as linear terms: only apply the logarithm transform to the GP ones;
- GP problems with posynomial (not monomial) equality constraints can sometimes be solved by relaxation: replace $h(x) = 1$ with $h(x) \leq 1$ and try to tweak the solution (especially if it is not unique).

Many problems can be approximated with GPs: a function f can be approximated by a monomial (resp. posynomial) if $F(y) = \log f(e^y)$ can be approximated by an affine (resp. convex) function. A *monomial fit* can be obtained by Taylor expansion, least squares, penalized least squares (L^1 regularization gives fewer terms), non-linear least squares for the relative error $|g(x_i) - f_i|/f_i$, or by minimizing the maximum

relative fitting error (after the logarithm transform, the problem becomes linear). A *max monomial fit* $f(x) = \text{Max}_k f_k(x)$ can be obtained by adapting the k -means algorithm (as a starting point, use k perturbed copies of a monomial fit). A *posynomial fit* can be obtained by non-linear least squares (Gauss-Newton, sequential quadratic programming).

Some generalizations are no longer convex:

- Signomial programming removes the constraint that the coefficients be positive: start with an initial guess, use a monomial approximation of the offending terms (perhaps with a constraint to force the solution to stay close to the current one); iterate;
- Mixed integer GP can be solved with heuristics (round the solutions, perhaps after tightening the constraints; or round one variable at a time, starting with those closest to an integer; or use branch-and-bound).

**Portfolio selection problems in practice:
a comparison between linear
and quadratic optimization models
F. Cesarone et al. (2010)**

The quadratic program with cardinality constraints

$$\begin{aligned} &\text{Minimize } x'Qx \\ &\text{such that } \sum x_i = 1 \\ &\quad x \geq 0 \\ &\quad |\text{supp } x| \leq k \end{aligned}$$

(polynomial if Q is positive definite, but NP hard in general) can be solved efficiently by noticing that the minimum is obtained in the interior of a face Δ_I of $\Delta = [\sum x_i = 1]$ where the restriction Q_I of Q is strictly convex; the solution is then $x_I^* = (\mathbf{1}'Q_I^{-1}\mathbf{1})^{-1}Q_I^{-1}\mathbf{1}$. We “just” have to find I such that $|I| \leq k$, $x_I^* \subset \Delta_I$ that minimizes $(\mathbf{1}'Q_I^{-1}\mathbf{1})^{-1}$. It turns out that, as we increase k , the corresponding I 's are included into one another: this gives an incremental algorithm to solve the problem. However, since there are many possible I 's at each step (as we increase k , they form a tree, and we want one of the longest branches), the complexity is still exponential in the worst case; one can try heuristics, e.g., keeping the solutions with the best values.

**Learning from data
Y.S. Abu-Mostafa (Caltech, 2012)**

Hoeffding's inequality measures how well in-sample results generalize out-of-sample: for proportions, on a sample of size N ,

$$P[|\mu_{\text{out}} - \mu_{\text{in}}| > \varepsilon] \leq 2 \exp(-2\varepsilon^2 N).$$

It also tells us if learning is possible at all: when you choose between M (independent) models,

$$P[|\text{Error}_{\text{out}} - \text{Error}_{\text{in}}| > \varepsilon] \leq 2M \exp(-2\varepsilon^2 N).$$

But M is usually infinite. In the case of classification, what matters is not the number of models, but the number $m(k)$ of different possible predictions they can make on the data: if there are k points, there are at most 2^k dichotomies but, in general, since the models have a specific form (e.g., linear), there will be fewer dichotomies. If the **VC (Vapnik-Chervonenkis) dimension** (intuitively, the effective number of parameters, or the effective degrees of freedom)

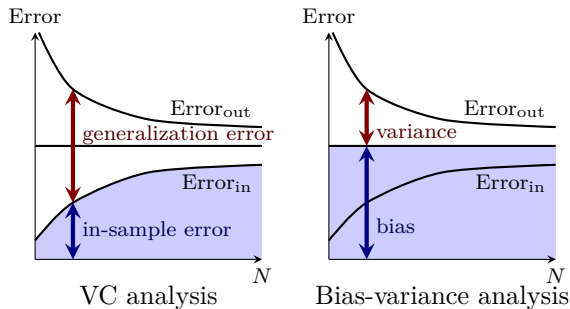
$$d_{VC} = \sup\{k : m(k) = 2^k\}$$

is finite, $m(k)$ is polynomial. We then have

$$P[|\text{Error}_{\text{out}} - \text{Error}_{\text{in}}| > \varepsilon] = O(N^{d_{VC}} e^{-\alpha \varepsilon^2 N}).$$

The multiplicative constant is too large for this inequality to be useful in itself, but it is believed that the order of magnitude is close to optimal. As a rule of thumb, you want $N \geq 10 d_{VC}$.

VC analysis can be contrasted with bias-variance analysis: the bias (sometimes called “deterministic noise”) is the error between the best approximation, in the set of models considered, and of the (unknown) data-generating mechanism; the variance is the error between the best approximation and that derived from the sample data.



To avoid overfitting, use regularization (add a $\lambda \|w\|^2$ penalty or, equivalently (by duality), a constraint $\|w\| \leq C$) and (10-fold) cross-validation (to select the regularization parameters and/or the model complexity).

The examples covered included singular value decomposition (SVD, for recommendation systems); the perceptron, logistic regression, neural networks; soft-margin support vector machines (SVM), with details of the corresponding quadratic optimization problem and its dual, which gives the support vectors (the number of support vectors measures the complexity of the model, not unlike the VC dimension); the kernel trick ($k(x, y) = (1 + x'y)^k$, which gives polynomials of degree up to k , $k(x, y) = \exp -\gamma \|x - y\|$); radial basis functions (apply k -means on the inputs, regress $y \sim \sum \exp -\gamma \|x - \mu_i\|$, choose γ and k by cross-validation – this is very similar to SVM with a gaussian kernel, where the μ_i are support vectors instead of k -means centers).

Introduction to logic

M. Genesereth and E. Kao (Coursera 2012)

Propositional logic is the logic of truth tables: given propositional constants p, q, r , etc. and a *truth assignment*, you can check if complex sentences ($p \wedge q$, $(p \vee q) \Rightarrow r$, etc.) are true or false. A sentence is *valid* if it is always true, for all truth assignments; *contingent* if it is sometimes true, sometimes false; *unsatisfiable* if it is never true; *satisfiable* if it is sometimes true (valid or contingent); *falsifiable* if it is sometimes false (unsatisfiable or contingent). A set of sentences Δ **entails** a sentence ϕ , written $\Delta \models \phi$, if, for all truth assignments for which Δ is true, so is ϕ . Entailment is a satisfiability problem: $\Delta \models \phi$ iff $\Delta \cup \{\neg \phi\}$ is unsatisfiable.

Satisfiability (SAT) is an NP-complete problem, but there are some improvements on the naive algorithm (checking all the truth tables): arrange the possible truth assignments in a tree (each level corresponds to a literal, each node corresponds to a partial truth assignment), simplify the problem at each node by partially evaluating the formula, and prune the tree whenever possible. Besides deterministic algorithms (DPLL), there are a few heuristics (WalkSAT).

The **Fitch system** is a set of rules of inference, to draw conclusions from premisses (notation: $\Delta \vdash \phi$). Since it allows subproofs

$$\frac{\phi \vdash \psi}{\phi \Rightarrow \psi}$$

(if you are not familiar with this notation: the premisses are above the bar, the conclusions below), its proofs are shorter than those of other systems (Mendelson, etc.). A proof system is *sound* when: if $\Delta \vdash \phi$, then $\Delta \models \phi$ (i.e., if something is provable, it is true); it is *complete* when the converse holds. Fitch for propositional logic is sound and complete.

Propositional resolution provides a more algorithmic way of checking entailment: distribute $(\phi \vee (\psi \wedge \chi)) = (\phi \vee \psi) \wedge (\phi \vee \chi)$ all the premisses to put them in *clausal form* $\bigwedge_i \bigvee_j p_{ij}$, often written as

$$\begin{aligned} &\{p_{11}, \dots, p_{1k_1}\} \\ &\dots \\ &\{p_{n1}, \dots, p_{nk_n}\} \end{aligned}$$

and apply the propositional resolution principle

$$\frac{\{p, q_1, \dots, q_m\} \quad \{\neg p, r_1, \dots, r_n\}}{\{q_1, \dots, q_m, r_1, \dots, r_n\}}.$$

The empty clause is a contradiction. The propositional resolution principle is not complete (e.g., $\{p\}, \{q\} \not\models \{p, q\}$), but unsatisfiable clauses can be proven to be so: to prove $\Delta \models \phi$, show $\Delta \cup \{\phi\} \vdash \{\}$.

Relational logic adds quantifiers, variables (needed if you want to use quantifiers), relations and functions. (You will often need an equality relation, syntactic sugar for an equivalence relation with the substitution property: $\forall x \forall y p(x) \wedge (x = y) \Rightarrow p(y)$.) We can still

define the notion of truth assignment and logical entailment but, in presence of functions, the set of sentences for which you have to provide a truth value (the **Herbrand base**) is infinite. Here are a few examples: **Z/4** can be described with 4 literals and relations “same”, “next”, “plus”; **N** (Peano arithmetic) can be described with one literal 0, a “successor” function and “same”, “next”, “plus” relations; lists can be represented with a “nil” literal, a “cons” function and an “append” relation; BNF grammars; propositional logic itself (*meta-level logic*).

Finite relational logic (*i.e.*, the Herbrand base is finite, in particular, there are no functions) is equivalent to propositional logic: to see it, write everything in *prenex form*, *i.e.*, with the quantifiers on the outside. It is *compact*: every unsatisfiable set of sentences contains a finite unsatisfiable subset. *Omega relational logic* (infinitely many literals but no functions) is not compact, but is *semi-decidable*: if a finite set of sentences is unsatisfiable, it can be shown in finite time (using the prenex transformation).

The Fitch proof system is still applicable to relational logic (with a few more rules: universal and existential introduction and elimination, for which you have to pay attention to free variables; domain closure; induction): if $\Delta \models \phi$, then there exists a finite proof $\Delta \vdash \phi$ (but if $\Delta \not\models \phi$, the search for a proof will take forever).

Relational resolution is useful, but incomplete: it is based on satisfiability, but relational logic is not decidable. To put an expression in clausal form, existential elimination, $\exists a p(x) \rightarrow p(a(y))$, adds a new function (*Skolem function*) of the enclosing universal variables y . Once you are only left with universal quantifiers, on the outside, you can drop them. Unification is also trickier: you may need to substitute some of the variables and drop part of the clauses.

Relational resolution can be used for *answer extraction*: to find x such that $p(x)$, add the clause $\{\neg p(x), \text{goal}(x)\}$, *i.e.*, $p(x) \Rightarrow \text{goal}(x)$, and continue until you get $\{\}$ (no solution), or $\{\text{goal}(a)\}$ (a is a solution), or $\{\text{goal}(a_1), \dots, \text{goal}(a_n)\}$ (one of the a_i is a solution). It may also take forever (relational logic is not decidable). The algorithm can be sped up by removing tautologies ($\{p, \neg p, \dots\}$), duplications (Ψ if $\Phi \sigma \subset \Psi$) or clauses containing a literal that is never negated.

In **Herbrand semantics**, the universe of discourse contains what the logic describes and nothing else; in **Tarskian semantics**, the universe of discourse can be larger (e.g., the reals and the hyperreals (non-standard numbers) are both models of the reals). Propositional logic and relational logic are examples of Herbrand semantics. **First order logic** is the tarskian equivalent of relational logic (you can emulate first order logic with relational logic: just add a new unary function). Fitch without domain closure and induction is sound

and complete for first order logic:

$$\begin{array}{ccc} \Delta \vdash_{\text{Fitch} \setminus \{\text{DC}, \text{IND}\}} \phi & \iff & \Delta \models_{\text{FOL}} \phi \\ \downarrow & & \\ \Delta \vdash_{\text{Fitch}} \phi & \iff & \Delta \models_{\text{RL}} \phi \end{array}$$

R in finance 2012

To design a real-time (*i.e.*, you cannot look into the future) finite-sample filter approximating some idealized filter,

$$\begin{aligned} y_t &= \sum_{k \in \mathbf{Z}} \gamma_k x_{t-k} \\ \hat{y}_t &= \sum_{k=0}^n \hat{\gamma}_k x_{t-k}, \end{aligned}$$

express it in frequency space

$$\Gamma(\omega) = \sum_k \gamma_k e^{-ik\omega}$$

and find the $\hat{\gamma}_k$ that minimize

$$\int_{|\omega|=1} |\Gamma(\omega) - \hat{\Gamma}(\omega)|^2$$

You can decompose this quantity into a penalty for the delay between the real and idealized filter and a penalty for the amount of remaining noise.

The singular value decomposition (SVD) can be used to quickly test for cointegration among a large number of assets.

The **d1m** package can fit, filter (Kalman), smooth state space models: ARMA, stochastic volatility, regression with time-varying parameters (**d1mModReg**), etc. and help you cross-validate the result.

The **Rcpp** package allows you to transparently manipulate R objects in C++, eschewing the gory macros you have to put up with when using **.Call** directly – and the **inside** package simplifies things even further. But you are left with those template-related error messages... It is even more useful with the advent of reference classes (R5). In the other direction, you can use **Rinside** to, say, include R in a GUI or a web application (e.g., with the Wt framework).

To mine textual data, use Python and NLTK, of the **tm** package and its many plugings (**tm.plugin.webmining**, **tm.plugin.sentiment**, **tm.plugin.tags**).

CppBugs is faster than **MCMCpack**, and (almost?) as flexible as **JAGS**.

Evaluating the design of the R language

F. Morandat et al.

Unflattering but objective evaluation of R: the language is designed to be used interactively (named and

optional arguments); the main data type is the vector (with support for missing values); it encourages vectorized operations. It mixes seemingly incompatible programming language paradigms: functional and lazy, but dynamic, imperative and reflexive (`parse`, `eval`, `quote`, `bquote`, `substitute`, `deparse`). The current implementation is “massively inefficient”, mainly because of the laziness of the language (which also fails to deliver the performance improvements it should bring), immoderate memory usage (garbage collection is more expensive, frequent and slow than it should). The two class systems (S3, S4 – there is also R5 and a few more in separate packages) look “like an afterthought”. It is “hopelessly non-thread-safe” and lacks standard data structures (growable arrays, hash maps). “It is not the ideal language to develop robust packages.” “As a language, R is like French; it has an elegant core, but every rule comes with a set of ad hoc exceptions that directly contradict it”.

The article also provides a *formal semantics* of the language, and analyzes large amounts of code, to see how the language is actually used.

The article does not mention the ability to easily describe statistical models (`formula`), the data manipulation (`reshape2`, `plyr`) and plotting capabilities – but these are add-ons, rather than parts of the core language.

The nature of alpha A.M. Berd (2011)

Even when they do not explicitly trade options, there is often a negative correlation between the returns of a hedge fund and market volatility, at least for “convergence” (mean-reverting) strategies: indeed, those strategies are essentially a short strangle $\mathcal{A}\backslash$. It is the opposite for momentum strategies. Measure this risk, diversify your strategies and, if it is not enough, add some explicit volatility hedging.

The feedback effect of hedging in portfolio optimization P. Henry-Labordère (2004)

Delta hedging can have a destabilizing market impact: to hedge a call, you buy when the market rises. This is the opposite of portfolio optimization, which suggests to buy when the price falls.

Circadian patterns and burstiness in human communication activity H.H. Jo et al. (2011)

Human activities (phone calls, tweets, etc.) show both periodicities (circadian, weekly) and bursts. The bursts can be modeled with a **cascading Poisson process**, *i.e.*, a Poisson process $(X_t)_t$ with intensity

$$\lambda_t = \lambda_0 + \lambda_1 \sum_{s < t} e^{-(t-s)/\tau} X_s.$$

The periodicity can be modeled with a change of time $Y_t = X_{T_t}$, with dT_t/dt periodic (and positive).

Robust pricing and hedging of double no-touch options A.M.G. Cox and J. Obłój (2009)

Given a set of calls and digital call prices, with no arbitrage opportunities (there are a few conditions to be satisfied: $C(0) = S_0$, $C(\infty) = 0$, $C'(0) \geq -1$, $D(k) = -C'(k)$), how much latitude do we have to extend the market model to include other options? In the case of double-no-touch options, one can devise a few super- and sub-hedges and show that the corresponding bounds are as tight as possible.

Conquering the Greeks in Monte Carlo: efficient calculation of the market sensitivities and hedge ratios of financial assets by direct numeric simulation M. Avellaneda and R. Gamba (2000)

When pricing options via Monte Carlo simulations, you do not get exactly the market prices; however, you can put weights on the sample paths to exactly recover them (use the maximum entropy principle to have uniquely defined weights). Those weighted paths can then be used to compute sensitivities (“Greeks”).

Vibrato Monte Carlo and the computation of Greeks S. Keegan (2008)

Option sensitivities (“greeks”) can be computed via

- Finite differences

$$\frac{\partial \text{Price}}{\partial \theta} = \frac{\text{Price}(\theta + \Delta\theta) - \text{Price}(\theta - \Delta\theta)}{2\Delta\theta}$$

- Likelihood ratio

$$\begin{aligned} \text{Price} &= E[\text{Payoff}] = \int \text{Payoff}(S) p(S) dS \\ \frac{\partial \text{Price}}{\partial \theta} &= \int \text{Payoff} \times \frac{\partial p}{\partial \theta} dS \\ &= \int \text{Payoff} \times \frac{\partial \log p}{\partial \theta} p dS \\ &= E \left[\text{Payoff} \times \frac{\partial \log p}{\partial \theta} \right] \end{aligned}$$

(but the variance is too high);

- Pathwise sensitivity

$$\frac{\partial \text{Price}}{\partial \theta} = E \left[\frac{\partial \text{Payoff}}{\partial \text{Spot}} \frac{\partial \text{Spot}_T}{\partial \theta} \right]$$

- Adjoint method: discretize the PDE

$$\begin{aligned} S_{n+1} &= S_n + a(S_n, t_n)h + b(S_n, t_n)Z_{n+1} \\ &= f_{n+1}(S_n, Z_{n+1}) \\ S_N &= f_N \circ \dots \circ f_1(S_0); \end{aligned}$$

we can compute $\partial S_N / \partial \theta$ from the f_i and the $\partial f_i / \partial \theta$ (not unlike the backpropagation algorithm, to compute the gradient of the loss function in a neural network), for $(Z_n)_n$ fixed; and then integrate Z out;

- Conditional expectation: simulate the price until the penultimate time step $S_{T-\Delta T}$, consider the gaussian distribution $S_T|S_{T-\Delta T} = S_T|Z_1, \dots, Z_{T-\Delta T}$, use the likelihood ratio method, and integrate Z out

$$\frac{\partial \text{Price}}{\partial \theta} = E_{Z_1, \dots, Z_{T-\Delta T}} E_{Z_T} \left[\text{Payoff} \frac{\partial \log p_{S_T|Z_1, \dots, Z_{T-\Delta T}}}{\partial \theta} \right]$$

- The Vibrato method decomposes this expression further.

Smoking adjoints: fast Monte Carlo Greeks

More details on the adjoint method to compute Greeks.

Web search queries can predict stock market volumes **I. Bordino et al. (2012)**

It could be useful as a warning signal, but they do not seem to check for confounding variables (e.g., the day of the week).

Pricing stocks with yardsticks and sentiments **S.M. Bustos et al. (2011)**

Another variant of the CAPM:

$$E[\text{return}_i] = \frac{\sigma(\text{return}_i)}{\sigma(\text{market without } i)} E[\text{market without } i]$$

rather than

$$E[\text{return}_i] = \frac{\text{Cov}(\text{return}_i, \text{market})}{\sigma^2(\text{market})} E[\text{market}].$$

Memory effects in stock price dynamics: evidence of technical trading **F. Garzarelli et al. (2011)**

Technical trading is thought to be a self-fulfilling prophecy; one can test how efficient it is as follows. A “support” (“resistance”) is a local minimum (maximum) in a window of width τ ticks; estimate the probability that prices rebound on (rather than cross) the latest support or resistance if they enter a band of width ε around them; fine-tune τ and ε to maximize this probability (and validate on a test dataset).

Improving recommendation quality by merging collaborative filtering and social relationships **P. De Meo et al. (2011)**

The user rating matrix R can be written as a product of the user preference matrix, and the item characteristic matrix, $R = PQ'$ (“non-negative matrix factorization”). The singular value decomposition (SVD) provides such a decomposition, but does not deal well with missing values. Tikhonov regularization can help:

$$(P, Q) = \text{Argmin} \|R - PQ'\|^2 + \lambda(\|P\|^2 + \|Q\|^2).$$

You can add another penalty

$$\mu \sum_x \sum_{\substack{\text{user } y \\ \text{neighbour of } x}} \|P_{x,\cdot} - P_{y,\cdot}\|$$

to account for social relationships.

Non-conservative diffusion and its application to social network analysis **R. Gosh et al. (2011)**

PageRank is the steady state probability distribution of a random walk on a network. The *alpha-centrality* can be defined in the same way, by using a non-conservative diffusion: the process does not move from one node to a single other node, but broadcasts to each neighbouring node, as an epidemic or a fad would do (the number of viruses or of infected nodes is not conserved) – this is not unlike the difference between deterministic and non-deterministic finite automata.

Information filtering via preferential diffusion **L. Lü and W. Liu (2011)**

Most recommendation algorithms are based on similarities between users or items or both. One could also use a diffusion process on the user-object bipartite graph.

Maximum entropy random walks in complex networks with limited information **R. Sinatra et al. (2011)**

You can construct a maximum entropy random walk on a graph using only local information: make the transition probabilities proportional to some power of the degree of the target node.

From brain to earth and climate systems: small-world interaction networks or not **A. Bialonski et al. (2011)**

The omnipresent “small world property” (the average shortest path length grows at most logarithmically with the number of nodes) could be an artefact of the way we sample real-world networks.

Rewiring world trade I: A binary network analysis **T. Squartini et al. (2011)**

Local properties (node degree) are sufficient to describe the international trade network (ITN, world trade web, WTW), when viewed as a non-weighted network, but not when viewed as a weighted (trade volume) network. To see it, you can build a family of randomized variants, by **local rewiring** and look at the distribution of some global quantities, such as degree correlation or clustering coefficient. There is a less computation-intensive method, using the maximum entropy probability distribution on the set of graphs satisfying the degree constraints.

The network of global corporate control
S. Vitali et al. (2011)

The presence of cycles in the cross-holding network of transnational corporations complicates the estimation of the control structure (who controls who). (It is not clear how they address the problem: they seem to just mitigate it.) The network has a bow tie structure: financial companies (left half) control transnational companies (knot), transnational companies have tight circular crossholding relations and control smaller companies (right half).

Propagation of cascades in complex networks: from supply chains to food webs
R.D. Smith (2011)

Modelling the bullwhip effect, on a graph (supply chain, food web), using birth-death processes.

The blogosphere as an excitable social medium: Richter's and Omori's law in media coverage
P. Klimek et al. (2011)

Word frequencies have statistical properties similar to earthquake energies, before and after an event.

$$\begin{aligned} w_i &\approx (t - t_0)^{-\alpha} && \text{before} \\ w_i &\approx (t_0 - t)^{-\beta} && \text{after} \end{aligned}$$

iSAX: indexing and mining terabyte-sized time series
J. Shieh and E. Keogh

To index large databases of time series, store the time series in a tree, whose nodes correspond to increasingly finer and/or more precise discretizations. This is not unlike wavelet coefficients, R-trees (used to index spatial databases) or k -d trees.

A decision-theoretic formulation of Fisher's approach to testing
K. Rice (2010)

Statistical tests can be interpreted in a decision-theoretic framework:

- A frequentist test minimizes the type II error rate (“inaccuracy”) under a constraint on the type I error rate (“embarrassment”);
- A bayesian test assigns a cost to type I and type II errors and minimizes the expected cost.

A gentle introduction to quantile regression for ecologists
B.S. Cade and B.R. Noon (2003)

Quantile regression addresses (and measures) heteroskedasticity. It can highlight changes in the distribution that are significant, but not visible on the mean. It allows you to focus on the tail, where interesting things happen. In ecology, it can help identify

limiting factors (corresponding to distribution truncation: $\bigwedge \mapsto \bigwedge_{\dots}$)

Machine learning markets
A. Storkey (2011)

There are many ways of aggregating statistical models (“ensemble methods”: boosting, mixtures, etc.). Here is another one, inspired by financial markets. Statistical models are agents in a market, each with a utility function, trading “bets” (options on the next data point). The price formation mechanism is similar to loopy belief propagation. Linear, logarithmic, exponential utility functions lead to the median, (weighted) mean or geometric mean of the models.

Value-at-risk in portfolio optimization: properties and computational approach
A.A. Gaivoronski and G. Pflug (2004)

The value-at-risk (VaR), computed from a set of scenarios, is neither smooth nor convex – there are local extrema everywhere. Since the VaR can be expressed as

$$\text{Max}_i^{k+1} f_i(x) \propto \sum_{\substack{I \subset [1, n] \\ |I|=k}} \sum_{i \in I} \mathbf{1}_{\{x: \forall i \in I f_i(x) \leq f_j(x) \\ \forall i \notin I f_i(x) \geq f_j(x)\}} f_i(x),$$

one can replace $\mathbf{1}_{\{x: f_i(x) \leq f_j(x)\}}$ with $\phi_\varepsilon(f_i(x) - f_j(x))$ for some smooth ϕ_ε approximating $\mathbf{1}_{\{x \leq 0\}}$. The resulting smoothed VaR (SVaR) is smooth and has fewer local extrema. (The authors do not seem to be worried by the size of the sum.)

Transaction costs, trading volume, and the liquidity premium
S. Gerhold (2011)

In presence of transaction costs, even if you want to hold a constant-weight portfolio, you cannot rebalance it continuously. This suboptimality corresponds to a liquidity premium:

liquidity premium \propto bid-ask spread \times share turnover.

(The liquidity premium is the difference in expected returns between a risky asset with transaction costs and a risky asset with the same utility but no transaction costs.)

Stochastic market efficiency
O. Peters and A. Adamou (2011)

In a stochastically efficient market, it is not possible to beat the market (in terms of expected growth rate) by holding the market portfolio and playing with the leverage: the optimal leverage, from the Kelly principle, is always 1. This notion is different from that of price efficiency, which is static (at one point in time).

How efficiency shapes market impact
J.D. Farmer et al. (2011)

Market impact is concave.

***Record statistics for biased random walks,
with an application to financial data***
G. Wergen et al. (2011)

Asymptotic behaviour of the record rate $P_n = P[X_n > \sup_{k < n} X_k]$ for a random walk with drift X .

Quantitative Trading
E.P. Chan (2009)

Elementary and superficial book on how to become an independent quantitative trader – in short: do not forget to backtest your strategies, with clean, survivor-bias-free data. The book ends with a list of possible strategies (earnings or macro announcements; gold vs gold miners pairs trading; Fama-French factor model; calendar trades: January effect, gasoline May futures in April, natural gas June futures from January to April) and a frighteningly incorrect interpretation of the p -value of a cointegration test.

Financial Risk Forecasting
J. Danielsson (Wiley 2011)

Clear book on the value at risk (VaR), at a very elementary level.

***Reconstruction of financial networks
for robust estimation of systemic risk***
I. Mastromatteo (2012)

The credit network between banks can be described by a liability matrix, L_{ij} indicating the funds lent by bank j to bank i .

Only some entries (those above a regulatory threshold, reported to a regulator) and the aggregate values (total debt and credit, from the balance sheet) are known, and the matrix is often reconstructed using a maximum entropy estimator, *i.e.*, by maximizing the Kullback-Leibler divergence between L and a uniform distribution:

$$\sum_{i,j} L_{ij} \log \frac{L_{ij}}{Q}$$

But this assumes that the credit relations are as evenly spread out as possible: the real credit network is more heterogeneous, the real liability matrix is sparse.

One can try to maximize the sparsity (number of zero entries) under those constraints, and then compute the corresponding maximum entropy matrix.

***Mean-variance portfolio optimization
when means and covariances are unknown***
T.L. Lai et al. (2011)

Mean-variance portfolio optimization assumes that the expected returns μ and variance matrix V are known: often, some estimators (sample estimators, factor models, shrinkage, Bayes, Black-Litterman, etc.) are simply plugged into the optimization problem. This 2-step procedure (estimation, then optimization) is suboptimal. One can replace the optimization problem

$$\text{Maximize } E[w'r_{n+1}|\mu, V] - \lambda \text{Var}[w'r_{n+1}|\mu, V]$$

with a stochastic optimization problem

$$\text{Maximize } E[w'r_{n+1}|r_1, \dots, r_n] - \lambda \text{Var}[w'r_{n+1}|r_1, \dots]$$

The corresponding “non-parametric empirical Bayes” frontier is higher than the shrinkage or bootstrap ones.

***Risk minimization in stochastic volatility
models: model risk and empirical performance***
R. Poulsen et al. (2009)

In a complete market with a risk-free asset B and a risky asset S , any European option X can be replicated with a self-financing strategy (α, β) .

$$\begin{aligned} X_t &= \alpha_t B_t + \beta_t S_t \\ dX_t &= \alpha_t dB_t + \beta_t dS_t \end{aligned}$$

If the market is not complete (e.g., a stochastic volatility model), you can still build a replicating strategy, but it will not be self-financing:

$$\text{Cost}_t = \int dX_t - \int \alpha_t dB_t - \int \beta_t dS_t$$

is not constant. Some people just use a strategy that would be self-financing in a complete market, delta-hedging, and hope for the best. Alternatively, one could try to minimize some measure of risk associated to this cost, e.g., $E[(\text{Cost}_T - \text{Cost}_t)^2 | \mathcal{F}_t]$. The problem need not have a global solution, but a local extremum is often sufficient.

***A fuzzy pay-off method
for real option valuation***
A. Collan et al. (2009)

A *fuzzy subset* A of X is a map $\mu : X \rightarrow [0, 1]$, interpreted as $\mu(x) = P(x \in A)$ (it corresponds to a probability distribution on $\mathcal{P}(X)$ with the property that $(x \in A) \perp (y \in B)$ whenever $x \neq y$). For instance, $\mu(x) = \delta_{x, x_0}$ corresponds to a single point and $\underline{\Delta}$ or $\overline{\Delta}$ are *fuzzy numbers*. They can be used, instead of probability distributions, as the payoff of an investment, when computing its value (via MCMC) as the risk-neutral expected present value.

***The US stock market leads
the federal funds rate and treasury bond yields***
K. Guo et al. (2011)

To find which, of two time series x and y , leads the other, find a path in their **recurrence plot** (the matrix of squared distances $d_{s,t} = (x_t - y_s)^2$) from one corner to the other, that minimizes the sum of squared distances. The algorithm can be made more robust to noise by replacing the dynamic programming equation

$$C_{s,t} = d_{s,t} + \text{Min}\{C_{s-1,t}, C_{s-1,t-1}, C_{s,t-1}\}$$

with

$$G_{s,t} = (G_{s-1,t} + G_{s-1,t-1} + G_{s,t-1}) \exp(-d_{s,t}/T)$$

leading to the “thermal optimal path” (T can be interpreted as a temperature).

Econophysics – complex correlations and trend switchings in financial time series
T. Preis (2011)

Pattern conformity measures how well we can predict the evolution of a time series for the next Δt^+ instants from the previous Δt^- instants, using pattern matching on previous intervals of size Δt^- (the patterns are normalized with their range, to account for changes in volatility). It can be normalized (in a complicated way) and plotted against Δt^- and Δt^+ . It is a non-linear measure of autocorrelation.

Pattern matching can also be used to study trends and crashes, on various time scales. Consider the time series between two local extrema (extrema on centered intervals of size $2\Delta t$), reparametrize them so that the time varies between 0 and 1, find matching time series, look at the volume or intertrade time time series for those matching price time series, and average them.

Switching processes in financial markets T. Preis et al.(2011)

Shorter article on the same subject.

Dynamic generalized Hurst exponent as a tool to monitor unstable periods in financial time series
R. Morales et al. (2011)

“Scaling behaviour” refers to the relation between the volatility of the returns on some asset and the horizon τ over which it is computed. The Hurst exponent H satisfies

$$\text{Volatility} \propto \text{Horizon}^H.$$

More generally, one can look at other moments:

$$q\text{th moment} \propto \text{Horizon}^{qH(q)}.$$

If the **generalized Hurst exponent** $H(q)$ depends on q , the times series is said to be **multifractal**.

The generalized Hurst exponent $H(1)$ can be computed on a moving window, with exponentially decreasing weights, and used as a bubble detector.

Similarly, one can study the tail of the returns distribution:

$$P[\log\text{-returns} \geq r] \underset{n \rightarrow \infty}{\propto} r^{-\alpha}.$$

Is there a bubble on LinkedIn’s stock price?
R. Jarrow et al. (2011)

One can apparently detect bubbles by checking if the squared volatility increases more than linearly with the price:

$$\int_a^\infty \frac{\text{Price}}{\text{Volatility}^2} < \infty.$$

(If this condition is satisfied, the price process is only a local martingale under the risk-free measure.)

How to detect an asset bubble
R. Jarrow et al. (2011)

More details about the (kernel-based) variance estimators used.

Why is order flow so persistent?
B. Tóth et al. (2011)

The **order flow** (signed traded volume, the sign indicating whether the trade was initiated by the seller or the buyer), computed from transaction data, is persistent over several days, and its autocorrelation can be decomposed into “order splitting” and herding components (you need to know which performed each transaction): order-splitting dominates, and anti-herding (from market-makers) becomes visible at longer time scales.

Investment volatility: a critique of standard beta estimation and a simple way forward
C. Tofallis

Should the usual beta

$$\begin{aligned} \beta &= \frac{\text{Cov}(\text{Portfolio}, \text{Market})}{\text{Var Market}} \\ &= \text{Cor}(\text{Portfolio}, \text{Market}) \times \frac{\text{Sd}(\text{Portfolio})}{\text{Sd Market}} \end{aligned}$$

be replaced by the simpler and more stable

$$\text{sign}(\text{Cor}(\text{Portfolio}, \text{Market})) \times \frac{\text{Sd}(\text{Portfolio})}{\text{Sd Market}}?$$

It corresponds to the slope of a line between the portfolio-vs-market and market-vs-portfolio regression lines, and minimizes the sum of the products of the deviations in the horizontal and vertical directions (triangle areas).

Hedging the smirk
D.S. Bates (2005)

Some of the greeks (Δ , Γ) can be computed without any model, only relying on the fact that option prices are homogeneous (of degree 1) in the underlying and the strike (actually, some models do not satisfy this property): indeed, from Euler’s theorem,

$$\text{Price} = \text{Spot} \frac{\partial \text{Price}}{\partial \text{Spot}} + \text{Strike} \frac{\partial \text{Price}}{\partial \text{Strike}}.$$

(For $\Gamma = \partial^2 \text{Price} / \partial \text{Spot}^2$, use the fact that $\partial \text{Price} / \partial \text{Spot}$ and $\partial \text{Price} / \partial \text{Strike}$ are themselves homogeneous.)

Since the Black-Scholes implied volatility fluctuates much less than the option prices (which move with the underlying), it can be used instead of the price, from

$$\begin{aligned} \text{Price} &= \text{BS}(\text{Strike}, \text{Volatility}) \\ \frac{\partial \text{Price}}{\partial \text{Strike}} &= \frac{\partial \text{BS}}{\partial \text{Strike}} + \frac{\partial \text{BS}}{\partial \text{Volatility}} \frac{\partial \text{Volatility}}{\partial \text{Strike}} \end{aligned}$$

***On the assessment of Monte Carlo error
in simulation-based statistical analyses***
E. Koehler et al. (2009)

Monte Carlo simulations provide noisy estimators, but many studies do not report the error coming from the finiteness of the sample – and many use a frighteningly small number of replications. To estimate this Monte Carlo error (MCE), plot the estimate versus the number of samples (ideally for several chains); bootstrap the Monte Carlo samples you have (bootstrap after bootstrap); you can also use the fact that $\text{MCE} \propto R^{-1/2}$ to justify the number of replications you chose.

***Factor analysis for multiple testing (FAMT):
an R package for large-scale significance
testing under dependence***
D. Causeur et al. (JSS 2011)

Knowledge of the dependency structure of the data can help improve the correction for multiple testing (FDR, etc.), which usually assumes the tests are independent.

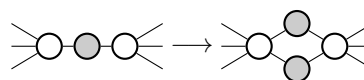
topicmodels: an R package for fitting models
B. Grün and K. Hornik (JSS 2011)

Latent Dirichlet allocation (LDA) models the occurrence of words as follows: for each topic, choose a word distribution (Dirichlet prior – the Dirichlet distribution is a distribution on $[\sum_i x_i = 1, \forall i x_i \geq 1]$); for each text, choose the distribution of the topics present in the text (another Dirichlet); to generate each word, select a topic from the topic distribution, and select a word from the word distribution of that topic. The correlated topic model (CTM) also allows for dependence between topics. Fit the model with Gibbs simulations or variational expectation maximization (VEM). In R, check the `lda` and `topicmodels` packages (Jags is more general, but slower).

***Consistently weighted measures
for complex network topologies***
J. Heitzig et al. (2011)

One often considers small finite “aggregated” approximations of large or infinite graphs, for instance, a finite mesh for the brain, finite approximations of climate networks whose vertices are points on the Earth and with an edge between two points if the the time series of (say) temperatures at those points are significantly correlated, network of IP address ranges when we are really interested in users, network of households to study people, of countries instead of consumers or companies, etc. Those aggregations or discretizations can be biased: for instance, a latitude-longitude mesh of the Earth gives undue importance to the poles. Graph-theoretic statistics (centrality, betweenness, etc.) have to be corrected for this bias: one can usually define weighted variants of those statistics by looking at how

they change under node splitting.



***Phase transition in the detection of modules
in sparse networks***
A. Decelle et al. (2011)

The formation of communities, in a graph, can be modelled with stochastic blocks: start with a set of nodes; assign each of the i to a group t_i ; add an edge between two nodes i and j with a probability proportional to the affinity $c(t_i, t_j)$ between their groups. The parameters of the model (number of groups, size of each group, edge probability) can be estimated via Monte Carlo simulation but, in the case of *sparse networks* ($O(N)$ edges in a network of size N , in particular, the graph is locally tree-like), belief propagation (BP, sometimes also called the “cavity method”) gives an asymptotically exact result in linear time.

***Infinitesimal methods
in Mathematical economics***
R.M. Anderson (2008)

Non-standard analysis for economists: ultrafilters, ultraproducts, hyperreal numbers, Loeb measure.

Programming a robotic car
S. Thrun (2011)

Presentation of a few algorithms you could use to build a self-driving car.

To estimate the position of a vehicle, you can start with an initial estimate (a probability distribution) and multiply it with the estimate from the measurement you have (this is the Bayes rule); when the vehicle moves, convolve this distribution with your estimate of the movement (this is the law of total probabilities). Iterate as the vehicle moves and you receive new measurements. The distribution of the location of the robot can be described by a discrete probability distribution (histogram filter), a sample of possible positions (particle filter) or a Gaussian distribution (Kalman filter – to understand where the Kalman filter formulas come from, just write the product and the convolution, first in dimension 1, then in general). The state space used includes position, orientation, and speed.

To plan the path the robot will follow, you can divide the world into a grid (or represent the possible paths as a graph, not necessarily a lattice) and use breadth-first search (from the starting point), dynamic programming (same idea, but from the goal) or the A* algorithm (similar to breadth-first search, but using a heuristic function that gives a lower bound on the distance, e.g., the Euclidian distance, ignoring obstacles). In the case of a car, the state space includes both position and orientation. You may want to smooth the path with an L^2 penalty: given the $(x_i)_i$, find $(y_i)_i$ to minimize $\sum_i \|x_i - y_i\|^2 + \lambda \sum_i \|y_i - y_{i-1}\|^2$.

The steering angle is often determined by PID control: a linear combination of the error (to put the robot back on track), its derivative (to avoid overshooting) and its integral (the total error so far, to take care of any systematic bias).

You can build a map using all the information accumulated so far (graph SLAM algorithm): the posterior distribution is a product of gaussians, but by expanding it, we end up with a sum of squares – getting to the estimate of the path of the vehicle and the position of the landmarks used is just a matrix inversion away.

***Constant-Q transform toolbox
for music processing***

C. Schörkhuber and A. Klapuri (2010)

On the implementation of the constant Q-transform (CQT), a wavelet transform whose frequency bins are geometrically spaced (12 to 96 bins per octave – the time-resolution of the bins is not constant, it depends on the frequency), well-suited to human music and auditory perception.

***Complex dynamics
in learning complicated games***

T. Galla and J.D. Farmer (2011)

Game-theoretic notions of equilibrium are not useful for complicated games such as financial markets, chess or go. If players learn their strategy via **reinforcement learning**, they oscillate along a chaotic attractor. While reinforcement learning may faithfully describe how people learn, it is inadequate for complicated games.

***Clarifications to questions and criticisms
on the Johansen-Ledoit-Sornette bubble model***

D. Sornette et al. (2011)

Review article on the log-periodic power law bubble model, with common misconceptions and implementation problems: it only models the expected log-price (not the log-price: there is a brownian motion in it); you should use the log-price, not the price; there are constraints on some parameters ($m < 1$, etc.); it only detects endogenous bubbles, not exogenous crashes; bubbles need not end with a crash, they can transition smoothly to another regime; the model is very sensitive to the start of the time window: try several, and look at the distribution of the results; use robust optimization algorithms (taboo search, genetic algorithms, etc.) and look at ensembles of solutions – there are many local extrema; be suspicious of solutions at the boundary of the search space; test your implementation on synthetic data, e.g., the fitted model plus AR(1) noise or reshuffled residuals – this gives confidence intervals for the critical time and tells you how robust to noise the model is; the output should be a distribution (an ensemble) of critical times, not a single value.

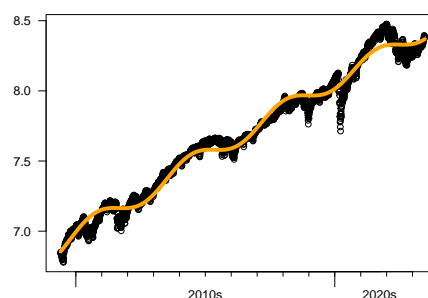
***A stable and robust calibration scheme
of the log-periodic power law model***

V. Filimonov and D. Sornette (2011)

Fitting the 7-parameter log-periodic power law (LPPL) bubble model

$$E[\ln \text{price}] = A + (t_c - c)^m (B + C \cos(\omega \ln(t_c - t) - \phi))$$

is not a 7-dimensional optimization problem: since 3 of the parameters are linear, you can vary the other 4 and compute those 3 via linear regression. Furthermore, by expanding the cosine, one can replace (C, ϕ) with $(C \cos \phi, C \sin \phi)$: there are then 4 linear parameters, and the quasi-periodic patterns in the cost function that were plaguing the optimization disappear.



***Detection of crashes and rebounds
in major equity markets***

W. Yan et al.

The LPPL model predicts the end of a bubble, but does not forecast what happens after: it could be a crash or just a regime change. For this, use pattern recognition (on the model parameters) and classification trees.

Role of diversification risk in financial bubbles

W. Yan et al. (2011)

The 3-factor Fama-French model (market, size, growth vs value) is sometimes replaced by a simpler 2-factor model, including the market and the **Zipf factor**, the difference in returns between the market portfolio and the equal-weighted portfolio. This factor can be added to the LPPL model.

$$E \left[\log \text{price} - \gamma \log \frac{\text{price}}{\text{equal-weighted price}} \right] = A + (t_c - c)^m (B + C \cos(\omega \ln(t_c - t) - \phi))$$

***Inferring fundamental value and crash
nonlinearity from bubble calibration***

W. Yan et al. (2010)

In the LPPL model, replace $\log p$ with $(p - p_0)^\gamma$, where p_0 is the fundamental value and γ the market over- or under-reaction. You can use statistical tests to compare this model with the LPPL one.

High frequency volatility
R. Almgren (2009)

Review of realized volatility estimators: $\hat{Q} = \sum (x_j - x_{j-1})^2$ is contaminated by microstructure noise – sub-sample, shift the window, and average.

**On covariance estimation
of non-synchronously observed
diffusion processes**

T. Hayashi and N. Yoshida (2005)

Given two diffusions x and y , observed at times t_1, \dots, t_n and u_1, \dots, u_m respectively,

$$\sum_{i,j} (x_{t_i} - x_{t_{i-1}})(y_{u_j} - y_{u_{j-1}}) 1_{[t_{i-1}, t_i] \cap [u_{j-1}, u_j] \neq \emptyset}$$

is a consistent estimator of their covariance

$$\langle x, y \rangle_{[0,T]} = \int_0^T \sigma_1 \sigma_2 \rho dt.$$

The corresponding correlation estimator is also consistent.

**A tale of two time scales:
determining integrated volatility
with noisy high-frequency data**

L. Zhang et al.

Observed high-frequency log-prices Y are contaminated with market microstructure noise ε :

$$Y_t = X_t + \varepsilon_t \\ dX_t = \mu_t dt + \sigma_t dB_t.$$

The integrated volatility $\langle X, X \rangle = \int_0^T \sigma_t^2 dt$ can be estimated with the realized volatility, *i.e.*, by summing the squared returns, either all of them, of those on a subgrid, to reduce the bias (Epps effect) introduced by the noise ε . Combining those estimators for two scales,

$$\widehat{\langle X, X \rangle} = [Y, Y]^{(n,k)} - 2 \frac{n-k+1}{nk} [Y, Y]^{(n,1)} \\ [Y, Y]^{(n,k)} = \frac{1}{k} \sum_{i=k}^n (Y_i - Y_{i-k})^2$$

is more efficient.

**Efficient estimation of stochastic volatility
under noise observations:
a multi-scale approach**
L. Zhang (2005)

Or more than two scales:

$$\widehat{\langle X, X \rangle} = \sum_k \alpha_k [Y, Y]^{(n,k)}.$$

**Estimating covariation: Epps effect,
microstructure noise**
L. Zhang (2009)

The previous-tick estimator of covariance (for asynchronous time series) is increasingly biased as the frequency increases (Epps effect): if is affected by discretization, non-synchronicity and microstructure noise. One can compute the optimal sampling frequency to minimize the mean square error (MSE) of the estimator, but 2-scale previous-tick estimators can help reduce the influence of microstructure noise even further.

**Assessing the performance of different
volatility estimators: a Monte Carlo analysis**
Á. Cartea and D. Karyampas (2012)

Volatility is often estimated from high-frequency time series by downsampling the data to remove the microstructure noise. Instead, one can model the price as a random walk (unobserved fair price), with added gaussian noise, and use a maximum-likelihood estimator (it can be implemented with a Kalman filter). Better, one can first identify and remove jumps.

**Jumps in financial markets:
a new non-parametric test and jump dynamics**
S.S. Lee and P.A. Mykland (2007)

To detect jumps, compare (look at the ratio of) the log-returns and the realized bipower variation (an estimator of the instantaneous volatility, consistent even in the presence of jumps)

$$\hat{\sigma}^2 = \frac{1}{k-2} \sum_j \left| \log \frac{S_{t_j}}{S_{t_{j-1}}} \right| \left| \log \frac{S_{t_{j-1}}}{S_{t_{j-2}}} \right|.$$

The test statistic is asymptotically gaussian in the absence of jumps, and a sum of a gaussian and (a random variable distributes as) the jump size in presence of jumps.

**Detecting jumps
from Lévy jump diffusion processes**
S.S. Lee and J. Hannig

To detect Lévy jumps (and not only large, Poisson jumps), replace the bipower variation with the truncated power variation

$$\hat{\sigma}^2 = \frac{1}{k} \sum_j \left(\log \frac{S_{t_j}}{S_{t_{j-1}}} \right)^2 1_{|\log S_{t_j}/S_{t_{j-1}}| \leq g}$$

(small Lévy jumps would bias the bipower variation). To test for small Lévy jumps, one can look for an excess of unusually large returns, when compared with the gaussian distribution. Those jumps cannot be identified (we can see that there are more large returns than expected, but we cannot say which come from the gaussian distribution and which are jumps), but one can compute a “belief” that a large return comes from a small Lévy jump.

Small, previously undetected Lévy jumps in stock prices can accumulate into previously puzzling detectable jumps in indices.

***Beta-arbitrage strategies:
when do they work, and why?***
T. Berrada et al. (2011)

A strategy long in low-beta stocks and short in high-beta stocks outperforms the market, in contradiction with the CAPM model. This can be explained with *stochastic portfolio theory*, i.e., the assumption that the market never becomes concentrated in a single stock (beta arbitrage is a form of diversity investing). The fact that some investors are leverage-constrained amplifies this phenomenon.

***A proof of the optimality
of volatility weighting over time***
W.G. Hallerbach (2012)

When investing in a 2-asset portfolio (one risky, one risk-free asset), adjust the weights so that volatility remains constant (use the implied volatility, e.g., the VSTOXX index, as a volatility forecast).

***Risk-based asset allocation:
a new answer to an old question?***
W. Lee (2010)

Review of those trendy risk-based approaches to portfolio construction: equally-weighted portfolio (very sensitive to the choice of the universe); mean variance portfolio (or equal marginal contributions to risk (MCTR)); most diversified portfolio ($\sum w_i \sigma_i / \sigma_{\text{portfolio}}$ maximal – but is only a differential diversification measure); equal risk contribution ($w_i \text{MCTR}_i$, also called parity portfolio: the problem is not convex and numerically challenging).

Liquidity of corporate bonds
J. Bao et al. (2008)

Illiquidity creates transitory price movements, and a negative autocovariance in price changes (at the transaction level). The opposite γ of that covariance can be used as a measure of illiquidity. If the bid-ask bounce is the only source of transitory price movements, the bid-ask spread is $2\sqrt{\gamma}$.

***Portfolio stochastic dominance
and bank liquidity risk***
S. Pagratis and N. Topalogou (2012)

A random variable X stochastically dominates a random variable Y if

$$\forall a \quad P(X > a) \geq P(Y > a)$$

or, equivalently,

$$\forall a \quad F_X(a) \leq F_Y(a).$$

The mean-variance portfolio optimization problem can be generalized by replacing mean-variance dominance with stochastic dominance (two gaussian random variables are stochastically comparable iff they have the same variance, and the order is given by their means). [I remain unconvinced: stochastic dominance is very, very rare: it is a partial order, and you could easily end up in a situation in which all or most portfolios are non-dominated – the efficient frontier would be too thick to be useful.] The portfolio optimization problem becomes: find λ to maximize

$$\max_z (F_0(z) - F_\lambda(z))$$

subject to

$$\forall z \quad F_0(z) \geq F_\lambda(z).$$

The article uses these notions to check if the balance sheet of banks (the assets are the loans and securities; the liabilities are the deposits, repos and off-balance-sheet commitments; the equity is the difference) is optimal, using the empirical cdf of the assets and liabilities.

***Anomalous price impact and the critical
nature of liquidity in financial markets***
B. Tóth et al. (2011)

Model to explain the square root in the price impact of large orders,

$$\Delta \text{price} \propto \sqrt{\text{size}},$$

which contradicts Kyle's linear model, based on the latent order book (for small orders, empirical evidence suggests $\text{size}^{0.2}$ or $\log(\text{size})$).

***A momentum trading strategy based on the low
frequency component of the exchange rate***
R.D.F. Harris and F. Yilmaz (2008)

Momentum trading strategies (monthly FX data) are often based on moving averages: this is suboptimal (newer observations should have more weight) and there is no objective way of choosing the parameters (it is either subjective or sample-specific). Instead, consider using kernel regression (sometimes called Savitzky-Golay filter, in digital signal processing (DSP)) or DSP analogues such as the **Hodrick and Prescott filter**

$$\text{Argmin} \sum_t (S_t - S_t^*)^2 + \lambda \sum_t (\Delta S_t^* - \Delta S_{t-1}^*)^2$$

(the first term ensures that S^* is close to the signal S , the second ensures smoothness) and just look at the slope of the smoothed signal.

Collaborative filtering with temporal dynamics
Y. Koren (KDD 2009)

Application of factor models (used to measure risk in finance) to recommender systems (Netflix, etc.).

Can we learn to beat the best stock
A. Borodin et al. (2004)

Cover's universal portfolios use frequent rebalancing to achieve large returns, especially if stock prices are mean-reverting, *i.e.*, if they have are negatively autocorrelated. They can be improved by exploiting cross-correlation.

Fiducial inference and generalizations
J. Hennig et al. (2009)

Fiducial inference is bayesian statistics with no prior: the model

$$\text{data} = f(\text{parameters}, \text{innovations})$$

can be rewritten as

$$\text{parameters} = f^{-1}(\text{data}, \text{innovations}).$$

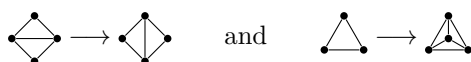
If you observe the data and know the distribution of the innovations, you have a posterior distribution for the parameters. (In general, f is not invertible, which complicates a lot of things.)

p-value precision and reproducibility
D.D. Boos and A. Stefanski (2011)

We tend to give a confidence interval to everything: why not for p -values, as well? A look at their standard deviations, bootstrap confidence intervals, and probability of reproducibility $P(p_{\text{new}} \leq 0.05 | p_{\text{old}} \leq 0.05)$ suggests that they are very imprecise: the order of magnitude (approximately, the stars displayed by most statistical software) is the best we can hope for.

Exploring complex networks
via topological embedding on surfaces
T. Aste et al. (2011)

Planar graphs are simpler than arbitrary graphs. Instead of trying to embed graphs in the plane, one can try to embed them in surfaces of genus g (for the lowest possible g). The maximal n -vertex graphs embeddable in a genus g surface can be described from two elementary moves:



Rationality, irrationality and escalating
behavior in online auctions
F. Radicchi et al. (2011)

Lowest unique bid auctions (people bid, much less than the price of the object; pay each time, much (100 times) more than the bid; are told if their bid is the highest or not; and the lowest unique bid wins) are all-pay auctions, lotteries decided by the outcome of a minority game: they are only profitable to the auctioneers.

Models for the impact of all order book events
Z. Eisler et al. (2011)

The mid-point price is a linear superposition of the impact of the previous prices, with decreasing weights (compute those weights from the autocorrelation function). This simplistic model can be improved by allowing for volume-dependent impact, different order types (market or limit), and weights that vary slowly with time.

Identification of clusters of investors
from their real trading activity in the market
M. Tumminello et al. (2011)

Investing styles (corresponding to the informed vs non-informed or fundamental vs technical oppositions in most agent-based market models) can be identified from transaction data as follows. Build a bipartite network, whose nodes are investors and days, with three types of edges, corresponding to “buy”, “sell” and “buy and sell later in the same day”; project it to a network of investors, with 9 types of edges, and weights corresponding to the number of days; to account for different trading activities, truncate the graph to a **statistically validated network**: compute the statistical significance of each edge, with a p -value, testing if the two investors act independently (given the number of orders of each type they send in the sample), adjusted for multiple tests (Bonferroni or FDR); finally, apply some community-detection algorithm such as infomap.

Life time of correlation between stock prices
on established and emerging markets
A. Buda (2010)

To find the “best” window size to compute a correlation matrix (between stock returns), check how the following quantities change: average number of consecutive days with a correlation above 0.5 (for a pair of stocks, or average for all pairs in a market); time in which half the connections in the minimum spanning tree (MST) have disappeared; average correlation (Epps effect).

A contextual risk model
for the Ellsberg paradox
D. Aerts and S. Sozzo

Quantum interpretation of knightian uncertainty (the future is random, but you do not know from which distribution it is drawn): quantum superposition corresponds to the use of mixture distributions as priors.

Asymmetric random matrices:
what do we need them for?
S. Drożdż et al. (2011)

Random matrix theory (RMT) can be generalized to study the distribution of the diagonal and off-diagonal elements, and the (complex-conjugate) eigenvalues of cross-correlation matrices $\text{Cor}(X, Y)$. The null hypothesis is still that (X, Y) is gaussian iid [I ini-

tially thought, from the title, that they were studying the eigenvalues of $\text{Cor}(X)$ where the distribution of (X_1, \dots, X_n) under H_0 is no longer invariant under the action of \mathfrak{S}_n .

Applications include the study of time series, $X = (X_1, \dots, X_n)$, $Y = (Y_1, \dots, Y_n)$ – but I am sceptical about the effect of autocorrelation (or serial dependence in general): with real-world data, we already know that H_0 is false.

***Endogenous bubbles in derivative markets:
the risk neutral valuation paradox***
A.F. Maccioni (2011)

In an incomplete market, the presence of risk-neutral investors (technical traders) lets the prices stray far away from the fundamental value of the assets. creating endogenous depressions or bubbles; when those traders disappear (or stop trading, for an instant), prices violently bounce back to the fundamental value.

Calculation of aggregate loss distributions
P.V. Shevchenko (2010)

(Good, simple introduction to Fourier methods for numeric computations in statistics.)

A **loss distribution** is the distribution of

$$Z = X_1 + \dots + X_N$$

where the X_i are iid and N is a random variable; we are interested in their value-at-risk (VaR) and expected shortfall (ES).

Analytically, it can be computed via convolution or characteristic functions

$$\begin{aligned} H(z) &= \sum_{k \geq 0} p_k F^{(k)*}(z) \\ \chi(t) &= \psi(\phi(t)) \\ h(z) &= \frac{2}{\pi} \int_0^\infty \text{Re}[\chi(t)] \cos(tz) dt \\ H(z) &= \frac{2}{\pi} \int_0^\infty \text{Re}[\chi(t)] \frac{\sin(tz)}{t} dt \end{aligned}$$

where H is the cdf of Z , F that of X , p_k the probability mass function of N , χ and ϕ the characteristic functions of Z and X , ψ the probability generating function of N . The moments of the compound distribution Z can be explicitly computed from those of X and N (to derive the formulas, use the characteristic functions).

Monte Carlo simulation is the easiest (and slowest) way of computing the VaR or ES of the compound loss (do not forget to estimate the error of those computations).

If the individual losses X_i have a discrete distribution, the distribution of the compound loss Z can be obtained recursively, in $O(n^3)$ time. If the probability mass function $p_n = P(N = n)$ satisfies

$$p_n = \left(a + \frac{b}{n}\right) p_{n-1}$$

for $n \geq 1$ (there are generalizations for $n \geq 2$), it can be computed in $O(n^2)$ (**Panjer recursion**). Discretization and underflow can pose problem. Without discretizing the losses, the Panjer condition gives an integral equation satisfied by h .

With the fast Fourier transform (FFT), one can compute the pdf h of the compound loss from its characteristic function χ . As with Panjer recursion, this still requires discrete losses. Aliasing error (artefacts at the boundary of the loss distribution) can be reduced by **tilting** the distribution, *i.e.*, by replacing $f(x)$ with $e^{-\theta x} f(x)$.

For the cdf H , one can use numerical integration methods (H can be expressed as an integral involving the characteristic function χ : there is no need to compute the pdf h), exploiting the oscillatory behaviour of the integrand.

Since the moments of the compound distribution are known, you can approximate the distribution with a simpler one (Gaussian, translated gamma) with the same moments.

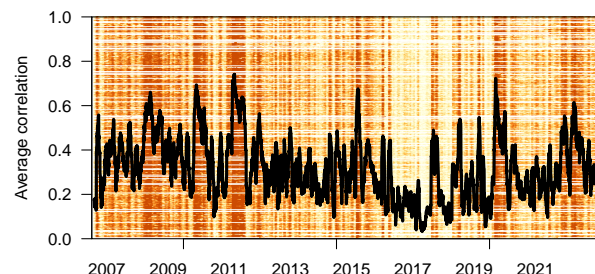
The asymptotic expansion

$$1 - H(z) \underset{z \rightarrow \infty}{\sim} E[N](1 - F(z))$$

provides a VaR estimator.

***Index cohesive force analysis
reveals that the US market became
prone to systemic collapses in 2002***
D.Y. Kenett et al. (2011)

The sample variance or correlation matrix of stock returns changes with time. To study those changes, you can look at the eigenvalues, their entropy, the average correlation, the average **partial correlation** (correlation without the market) – the ICF is the ratio correlation over partial correlation. A rasterplot of the average (partial) correlation by stock can help identify regime changes: the partial correlation became negligible in 2002.



Detecting novel associations in large data sets
D.N. Reshef et al. (2011)

The MIC (maximal information coefficient) is a measure of dependence defined (computationally) as follows. Divide the data into an $n \times m$ grid, discretize the data along this grid, compute the mutual information,

normalize it (divide by $\log \text{Min}(x, y)$); find the division that maximizes the mutual information; repeat for all values of n, m with $nm \leq N^{0.6}$ (where N is the number of observations). The MIC is the maximum value in this matrix.

It can be interpreted as a generalized correlation coefficient and can identify non-functional relations (a circle, etc.), but is misled by some patterns – for instance, both a line and a 2×2 checkerboard pattern give the maximum value. (Also, its power is apparently much lower than that of the correlation distance.)

From the matrix M of mutual information, one can derive other measures: $\text{Max}|M_{ij} - M_{ji}|$ measures monotonicity, $\text{MIC} - \text{Cor}^2$ measures non-linearity.

Brownian distance covariance **G.J. Székely and M.L. Rizzo (2009)**

The distance covariance between two random variables X and Y with characteristic functions ϕ_X and ϕ_Y is

$$\int w(x, y) |\phi_{X,Y} - \phi_X \phi_Y|$$

with $w(x, y) \propto |x|^{-p-1} |y|^{-q-1}$ (p and q are the dimensions of X and Y).

If U is a stochastic process on \mathbf{R} , let $X_U = U(X) - E[U(X)|U]$ be the U -centered version of X ; in particular $X_{\text{id}} = X - E[X]$. The **brownian covariance** is defined by

$$\text{Cov}_{W,W'}^2(X, Y) = E[X_W X'_W Y_{W'} Y'_{W'}]$$

where X' and Y' are iid copies of X and Y , and W, W' are independent brownian motions. It coincides with the distance covariance. The result can be generalized to Lévy fractional brownian motions of Hurst index $H \in (0, 1)$ and $w(x, y) \propto |x|^{-p-2H} |y|^{-q-2H}$.

Leverage aversion and risk parity **A. Asness et al. (2011)**

There are many reference portfolios: capitalization-weighted (the common approximation of the market portfolio), equal-weights, fundamental weights (since the market portfolio overweights overvalued stocks, one can try to replace the stock price with some “fair value” derived from the balance sheet – it is an accounting-motivated way of shrinking the capitalization-weighted portfolio towards the equal-weighted one), minimum variance, tangent (the CAPM theorem claims it is the market portfolio). The risk parity portfolio is yet another (robust) approximation of the elusive market portfolio, using equal contributions to risk. This is theoretically justified if some investors are averse to leverage: they would overweight risky assets (stocks) to achieve higher returns, increasing demand for riskier assets and lowering their price. Historically, a 170% bond, 30% stock portfolio performed better than the market portfolio and the classical 60% bond, 40% stock one.

Counter-point to risk parity critiques **E. Peters (2010)**

A few caveats on the implementation of risk parity strategies: avoid leveraging assets that are already internally leveraged (e.g., stocks, whose debt/equity ratio is often 2/1); hedge inflation; do no leverage asset classes with no expected returns, such as commodities – only use them to hedge inflation.

The hidden risks of risk parity portfolios **B. Inker (2010)**

Remember that the standard deviation does not measure all the risk, especially for assets with skewed returns – and leverage increases the danger.

Financial Theory **J. Geanakoplos** **Yale (2009)**

Some universities have started to put some of their undergraduate (*i.e.*, elementary) courses online: this one is an introduction to finance, describing models, examples or experiments to illustrate general economic and financial facts.

Economy

Many price-setting mechanisms (seller-derived (supermarket), haggling, price regulation, *tatonnement*, *pit* (like haggling, but with several buyers and sellers), bid/ask (as in computer-trading), specialist, etc.) lead to the same price (this is the law of one price), the price predicted by the law of supply and demand. This can be seen in the following experiment: take 10 students, 5 buyers and 5 sellers, assign a preferred price to each, ask them to trade using the “pit” method (shout the desired price, above their preferred price for seller, below for buyers, and modify your offer until you find a counterparty). Even if it were theoretically possible for everyone to find a counterparty with a profit on both sides, this is not what happens: the final price p is such that the number of sellers below p equals the number of buyers above p – the others could not find a counterparty.

In this experiment, the average price (for buyers, or for sellers) did not play any role: only the price at the margin mattered. The price is decided at the *margin*: the market price is the reservation price of the marginal buyer or seller, *i.e.*, the price for the next person to buy or sell. This also explains Adam Smith’s water and diamonds paradox: water is useful but cheap, while diamonds are useless but expensive. Price and value are not the same thing.

Economic models are usually defined with a set of agents, each with an initial endowment and a utility function, trading with one another to change their allocation and maximize their utility function. It was once thought that the resulting equilibrium was maximizing the sum of the utilities – this only happens in the unrealistic case where the marginal utility is constant. The equilibrium is just Pareto efficient: changing the

allocations would make someone worse off (but there are many Pareto-efficient allocations).

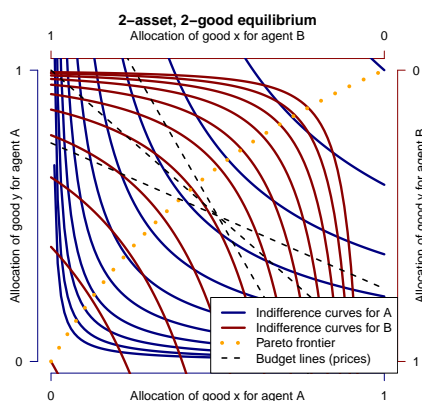
To find the equilibrium, just write the following equations:

- Markets clear, *i.e.*, supply equals demand, *i.e.*, for each good, the sum of the initial endowments equals the sum of the allocations;
- All the money is spent, *i.e.*, for each actor, the value of the initial endowment is the value of the allocation;
- Everyone maximizes their utility: the Lagrange multipliers conditions can be written as

$$\frac{1}{p_x} \frac{\partial U_A}{\partial x} = \frac{1}{p_y} \frac{\partial U_A}{\partial y}$$

for all goods x, y , and all actors A (U' is called marginal utility).

The prices are only determined up to a multiplicative constant. There is no “just” price: it depends on utilities and endowments.



Assets and time

To jump from economic models to financial models, we need to add time, assets and (later) risk.

Adding time is easy: let x and y be the same good (say, apples), but x is now and y next year. For instance, if the utility is $\log x + \frac{1}{2} \log y$, we are impatient, we value apples more today than next year, *i.e.*, we discount apples in the future.

We can also add assets: for instance, a piece of paper giving you the right to receive one apple today x and one apple tomorrow y is an asset – an apple tree is also an asset, with the same dividends $(1, 1)$.

The situation may look complicated: we have endowments of x, y , of the asset $(1, 1)$ (and of other assets, defined by their payoffs or dividends); we have the utility for each actor; we are looking for the price of each good and asset, and the allocation of each good and asset. But it can be simplified by just removing redundant assets or goods: assets such as $(1, 1)$ are just linear combinations of goods.

With the introduction of time, the notion of price becomes trickier. The prices are expressed in some currency. In the 1-period set-up, we could “normalize” the prices by assuming that the price of good x was 1 currency unit. In the 2-period setup, we can do exactly the same: assume that the price of good x is one currency unit, and look for the price of good y (the same asset next year) in the same currency – the currency *this year*. In the real world, the market price we would see would be expressed in a different currency: the currency *next year*. The difference (strictly speaking, the ratio, expressed as a rate) between the currency this year and the currency next year is the inflation. To model it, we would need to add a “theory of money” (how much money there is in the economy, how quickly it circulates, etc.) to our model. The difference between the price this year (in today’s currency) and the price next year (in today’s currency) is the **real interest rate**. This is what we are interested in. The price, in today’s currency, of next year’s good, is called its present value. The difference between the price this year in today’s currency and the price next year in next year’s currency is the nominal interest rate: it combines (muddles) both elements. We will only use today’s currency.

$$1 + \text{real interest rate} = \frac{1 + \text{nominal interest rate}}{1 + \text{inflation}}$$

The real interest rate is usually positive. It can be negative when the government tries to boost demand: it is then preferable to spend now, and even borrow to spend now, rather than wait until next year. Interest is “crystallized impatience”: we prefer goods today rather than tomorrow. It may also include differences in supply between today and tomorrow.

Bonds and rates of return

Bonds (coupon, zero, annuity, perpetuity) can be used to spread expenses over time. For instance, if a university needs 100,000,000 per year, for the next 10 years, to repair badly maintained buildings, it does not have to cut its budget by that much: if the cost can be spread over the life of the university, *i.e.*, if they are willing to pay forever, they can replace the costs with a perpetuity with the same present value (PV) and reduce the annual cost:

$$\text{PV}(10\text{-year } 100 \text{ million annuity}) = \text{PV}(\text{perpetuity}).$$

(You have to choose an interest rate for this, e.g., 5%.)

The notion of present value can also be used to measure the performance of a hedge fund: since the cash flows and the assets under management are irregular, the notion of average return can be contentious. The geometric average of the annual total returns is a biased measure: it gives the same importance to years with little capital (and amplifies little profits) and less importance to years with more capital (and reduces losses, e.g., caused by capacity constraints). The **internal rate of return** (IRR, sometimes called yield-to-maturity), *i.e.*, the constant rate for which the present value of all

the cash flows (both positive and negative) is zero, is a better alternative, though plagued by its own problems. It is also biased, because clients tend to leave after a bad year (so that there is more weight on those bad years) and invest after good years (so that there is less weight on those good years). If there is a gap, a long series of zeroes, in the stream of cash flow, everything that comes after will be discounted a lot and will have a negligible effect. The IRR is not always well-defined: for instance, $(1, -4, 3)$ could suggest either 0% or 200%. The IRR assumes that the interest rate is deterministic and constant (we are throwing away relevant information: we know the risk-free discount factors).

There is also a (uniquely defined) modified internal rate of return (MIRR). The IRR assumes that you can reinvest the money at the IRR rate, while the MIRR uses a reinvestment rate: compute the future value of the inflows, after n periods, with the reinvestment rate; compute the present value of the outflows (negative cashflows), now, with the financing rate; convert into a rate. In R, check `financial::cf`. (The classical alternative to the IRR is the present value of the stream of cash flows for the risk-free interest rate.)

Some people use the “current yield” (the bond rate divided by price) instead, especially if they want to sell you something: this is “the wrong way of measuring things, which can get you confused”. (Be wary of anything called “yield”: there are many, many definitions, and they are almost all misleading.)

Using risk-free bonds of several maturities one can compute the price today of \$1 in n years (that is a **discount factor**), and the **forward rates** (*i.e.*, the rate between years $n - 1$ and n : it would be the future interest rate, if it was deterministic and known – it is not). Some also compute the yield (and plot the yield curve), but that is misleading: it is the annual rate over those n years computed assuming that the interest rate is constant – but we look at the yield curve because it is not constant. (The yield is not unlike the implied volatility and the smile: it is computed under the assumption that it is constant, and we look at how it varies...)

Profit is sometimes incorrectly measured as the cash flow you receive after one period, forgetting the change in present value. In times of crisis, when the value of the firm drops, many banks and businesses report the cash flow instead of the profit, to hide the losses. This kind of deception is very common with bonds: this is how the “current yield” is defined.

$$\text{Profit} = \frac{CF_1 + PV_1 - PV_0}{PV_0} \neq \frac{CF_1}{PV_0} = \text{Current yield}$$

Carry trades use this idea to hide losses: for instance, if the 2-year forward interest rate is 2% and the 5-year 5%, you can buy a 5-year bond (you pay 100 and receive 5, 5, 5, 5, 105) and sell a 2-year bond (you receive 100 and pay 2, 102). More generally, a carry trade is a portfolio of two bonds, one long, one short, with a

positive cash flow at the beginning and a negative cash flow at the end, so that the portfolio value be zero (easy to do if the yield curve is not flat). If you only report the cash flow and not the present value of the outstanding bonds, it looks positive. That is why you should **mark-to-market**.

Social Security

Social security (an insurance against living too long) works as in the following parable: a son gives \$100 to his father (say, for a life-prolonging operation), later asks his own son to pay back the \$100 that helped his grand-father live a few more years, this son then asks his own son, and so on – each generation contributes the same nominal amount, which contributes to the PV of the first father, *e.g.*, $PV = 50 + 25 + 12.5 + \dots = 100$. (In case of a baby boom, the amount to repay (per person) decreases.) In other words, social security is a (beneficial) Ponzi scheme – money is another widespread Ponzi scheme.

The **overlapping generations model** can explain and quantify what happens. Generation n lives in periods n and $n + 1$, with endowments $(3, 1)$, *i.e.*, 3 units when young, 1 when old. Generation 0 only has a $(0, 1)$ endowment, but also owns land, an asset with payoff $(1, 1, 1, \dots)$: in period 1, they will consume the good produced, sell the asset to the next generation, and buy more good (from the next generation) with the proceeds. The utility is $\log x_t + \log x_{t+1}$. One can compute the equilibrium.

Social security worsens this situation: the $(3, 1)$ endowment is replaced by $(1, 1)$. This is a huge benefit for the first generation, but every subsequent generation has to pay the price; the cost does not fall, and remains the same generation after generation, because of the interest rate.

The model can be made more realistic by considering a growing population, growing land dividends, *etc.*: the situation improves, but only slightly – the interest rate would just be higher.

Some economists claim that the baby boom should not have any effect on the stock market, because we know in advance how many people will retire, and this information is already in the prices. However, the interest rate fluctuates with the population: this is what impacts the stock markets – demography has an effect on stock prices. If we compare the utility of the generations, we find that the utility of baby boomers is lower.

(The overlapping generations model for social security can be augmented with random stock dividends.)

Social security can be fixed as follows:

- Separate the legacy debt from social security;
- Create a new security, an annuity indexed on the average wage: its performance will track that of the stock market, but with fewer, slower fluctuations; the market will price it;
- If you want some redistribution, by paying 10% to

14% of your income (depending on how much you earn), 12% would go in your social security account.

Uncertainty

To fully leave Economy for Finance, we need to add uncertainty to our models: the future state of the world is a random variable, but with a known distribution (we exclude knightian uncertainty, *i.e.*, uncertainty about the probability distribution).

Uncertainty can have a surprising effect on the discount factor. We see a big difference between today and tomorrow, but very little between one year and one year and one day. But the discount factor is supposed to decrease exponentially. This can be explained if the interest rate is unknown, stochastic, and (unrealistically) follows a geometric Brownian motion: the expected discount factor decay (asymptotically) in $t^{-\alpha}$ (**hyperbolic discount**).

Evidence shows that prices already include a lot of information. For instance, prices of orange juice (concentrated, mainly from Florida) include weather information; weather forecasts cannot help forecast prices, but prices can help forecast weather. The **rational expectations** assumption goes further and claims that the current price is the expectation of the present value of the future price (this is not entirely true: we will see later how risk changes prices – the difference is the risk premium).

$$\text{Current price} = E[\text{PV}(\text{future price})].$$

From bond prices, one can estimate the probability of default of a country or company (just compare the price with a risk-free bond, *i.e.*, compute the default spread).

$$\begin{aligned} \text{Bond price} &= \frac{1 - P(\text{default})}{1 + r} \\ &= \frac{1}{(1 - \text{default spread})(1 + r)} \end{aligned}$$

With several maturities, one can check how this probability changes with time. The CDS contains the same information.

Shakespeare's play *the Merchant of Venice* is about **collateral** to control default risk (people not keeping their promises): a defaulted loan makes you lose "a pound of flesh", choosing the wrong casket forces you to remain single, losing the ring entails your death.

In the *black-and-red game*, you draw cards from a 52-card deck, receive \$1 for each red card, pay \$1 for each black card and stop when you want (in particular, you cannot lose: if you wait until the end of the deck, you are flat). What is the value of this game and what is the "optimal" (maximum expected payoff) strategy? This **optimal stopping time** problem can be solved via dynamic programming ("backward induction"): let $V(a, b)$ be the expected value of the game if there are a red and b black cards left, and compute it recursively.

$$V_{a,b} = \left(\frac{a}{a+b}(1 + V_{a-1,b}) + \frac{b}{a+b}(-1 + V_{a,b-1}) \right)_+$$

Callable bonds are another optimal stopping problem (use, e.g., a geometric brownian motion for the interest rate and compare the price of a bond and a callable bond; use a binomial or trinomial (recombining) tree): the borrower has the option to prepay the bond at any time. Mortgages are a special case.

Mortgages: history

Mortgages have been around since Babylon (1000 BC). They used to be coupon bonds (say, 7 7 7 7 7 ... 107), with a "balloon payment" at the end – but people would often default just before. After the 1929 crisis, amortizing mortgages, with identical dividends (e.g. 8 8 8 ... 8) became more popular: the present value (or "remaining balance"), and the risk for the bank, decreases with time.

Then, Freddie and Fannie pooled mortgages and sold them: the risk was limited (the loans were standardized, with stringent conditions on the borrowers), distributed (loans from different regions were pooled), and securitization made mortgages liquid (you could easily resell your Freddie or Fannie shares).

Then, CMO (Collateralized mortgage obligations) cut the pool into pieces: for instance, floaters (constant cash flow plus interest rate), reverse floaters (constant cash flow minus interest rate) and a residual piece for the defaults and the prepayments. There were arbitrage opportunities when the price of the pieces did not match the price of the pool. Hedge funds were selling the riskiest parts and hedging the rest; other investors were selling the easy parts and worrying about the risky parts remaining on their balance sheets. However, the chain from home owner to mortgage pool to CMO to investor was getting longer, and the same collateral was re-used at each step.

Then, new mortgage markets appeared, besides agency (Freddie, Fannie) mortgages: jumbo prime (large mortgages, that were previously excluded); alt-A (almost prime); subprime; uncollateralized.

Then, mortgage pools were cut into pieces (bonds): AAA, AA, A, BBB (the defaults and prepaes are put into the lowest piece first, until it is full). CDS (credit default swaps), insurance on each of those bonds, also appeared.

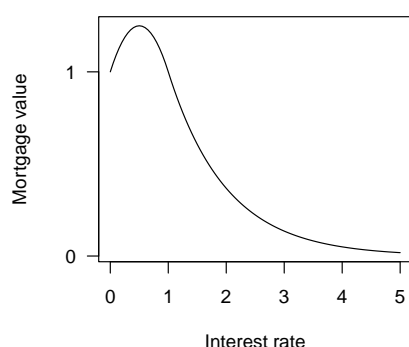
Then, CDOs appeared (take many BBB pieces, pool them together, and split them, into AAA, etc.), and CDO squared (do it again, take many AA CDOs, pool them together, and split them).

Mortgages

Mortgages are just callable bonds, and computing their prices should be an optimal stopping problem. However, borrowers do not prepay their mortgages optimally: banks estimate the proportion of people who will prepay in order to lower the cost of the mortgages and be more competitive – for the borrower, there can be an actual arbitrage opportunity. Modelling the prepay rate as a function of the interest rate poses a survival problem: people who have prepaid are more alert

or attentive than those who remain: the more people prepay, the less likely the remaining ones are to prepay. Agent-based models work better: Each agent has a “prepay cost” (that depends on the credit rating of the borrower) that measures how attentive he is to the interest rate, and acts rationally wrt this cost; one can calibrate the distribution of this prepay cost with the data. Other parameters can be added: attentiveness, value of the mortgage, location, pay of the borrower, etc.

The value of the mortgage depends on the interest rate, as an option does, but the shape of the payoff is tricky, dangerous, when the interest rates are low – that part should be carefully hedged.



Hedging

A **hedge fund** is an asset manager that uses leverage, hedging, is lightly regulated (*i.e.*, the clients are sophisticated investors) and has high fees. Hedge funds typically go bankrupt when they do not hedge properly or encounter liquidity problems (e.g., margin calls).

Here is a hedging example. Someone is willing to bet that team A will win at 60/40 odds, while the bookmakers offer 50/50: how do you profit from this discrepancy of views? You can accept the bet, but place an opposite bet with the bookmakers, for a carefully chosen value, so that you have a sure profit in all cases – there are no probabilities involved.

When time is involved, **dynamic hedging** may be required. For instance, imagine someone is willing to bet that team A will win over team B, over the next 10 matches, at 60/40 odds, but the bookmakers offering 50/50 odds only allow you to bet on individual matches: how do you profit? The solution is similar, but you have to change your bet after each match.

Dynamic hedging also occurs in finance: for instance, if the market price of a mortgage is lower than what your model predicts (e.g., because the market does not account for prepayments), but is extremely sensitive to interest rates, you can hedge interest rate fluctuations away.

Everyone wants to hedge, but some risk will remain (markets are not complete).

Earlier, we have claimed that the price was the expected payoff: it is actually less, to compensate for the risk. The difference is the **risk premium**. [What we really did was use the present value of the future cash flows as utility, maximize expected utility, and compute the corresponding price: that kind of utility is unrealistic, but the rest of the reasoning is correct.]

Utility should be concave, to account for diminishing utility, *i.e.*, risk aversion – this is Jensen’s inequality: we prefer \$1 for sure to \$1 on average. To avoid infinite expected utilities, you may also want utility to be bounded (for instance, how much would you pay for the game: “flip a coin until you have tails and receive 2^n (or 2^{2^n}) dollars, where n is the number of tosses needed”?).

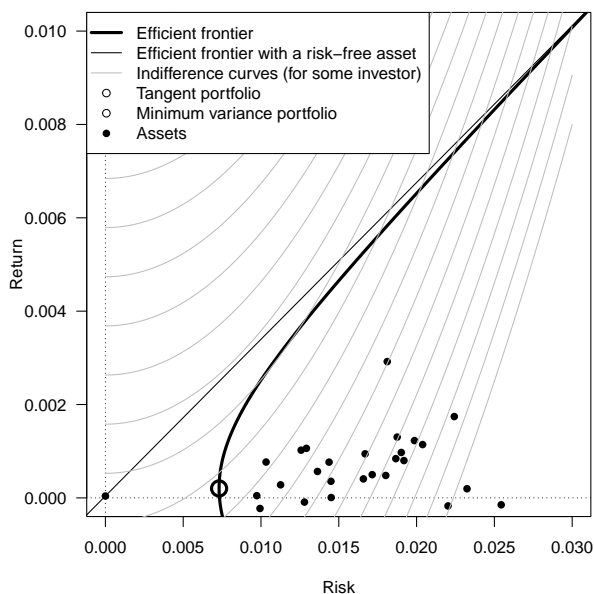
With quadratic utility, $U(x) = ax - x^2$, the expected utility only depends on the mean and variance of x .

An **Arrow-Debreu security** is an asset that pays \$1 in one state of the world, and \$0 elsewhere. CDS and binary options are almost Arrow-Debreu securities: they are event-dependent, not state-dependent. In a complete market, the price of an Arrow-Debreu security is proportional to its probability.

The **mutual fund theorem** says that in an incomplete market, everyone diversifies by holding the aggregate economy and cash: the only thing that differentiates economic actors’s allocations is the proportion of cash.

This assumes that all investors are rational, have access the the same information and investment opportunities, know the correct expected returns and variances, and have quadratic utility (but not necessarily the same utility). Of course, we do not know what the “market” portfolio – but it is larger than we think.

The advantage of diversification can be seen with two assets: if $\text{Cor}(X, Y) < 1$, the variance of $Z = \lambda X + (1 - \lambda)Y$ can be lower than that of X and Y (try to plot those random variables, as λ varies, in the $\sigma(Z) \times EZ$ plane).



The Sharpe ratio of an asset or portfolio is the slope of the line going through it and the risk-free asset: $(EX - r)/\sigma(X)$. You should maximize your Sharpe ratio. A stock is worth more if it increases the Sharpe ratio. This can be quantified by the **covariance pricing theorem**:

$$\text{Price} = (1 + r)^{-1} EX - \alpha \text{Cov}(\text{Market}, X),$$

i.e., the risk is *not* measured by the variance, but by the covariance. The price is determined by the marginal utility, by the utility of a marginal buyer, by how much diversification the asset adds to the market portfolio – not by something intrinsic. (This still assumes quadratic utility.) This can be somewhat unintuitive. For instance, if GE and a startup have the same expected returns, which has the highest price? You might think it is GE, because it has a lower risk, as measured by the variance. It is actually the startup, because it has a lower risk, as measured by the correlation with the market.

In particular, $(EX, \text{Cov}(\text{Market}, X))$ should be on a line.

The CAPM worked well for 20 to 30 years, but no longer works. The same goes for the Black-Scholes model: the Gaussian daily returns assumption was almost valid in the 1970s, but no longer is.

Leverage cycle

The **leverage cycle** can be modelled as follows: two assets (gold, risk-free, and an oil well, with an uncertain payoff), two states, many actors, with identical endowments (1,1) but heterogeneous beliefs on the probabilities of those states. Actors can borrow gold using oil wells as collateral from less optimistic investors, with various leverages. There is not a single supply and demand equation, but many, one for each possible value of the leverage. (It turns out that only one of those loans is actually traded: the leverage is uniquely defined.) The availability of leverage increases the price.

Crashes appear in the 2-period model: after a bad news, the most optimistic investors are bankrupt, the leverage drops, marginal buyers are less optimistic and pay less, and the price drops.

This is what happened with the subprime mortgage crisis. The apparent risk was low because the securitized mortgages were collateralized (by the houses) and investors were also buying insurance, in the form of CDSes, from insurance companies. However, those CDSes were not collateralized. If the BBB mortgage bonds went, so would the CDSes, CDOs and CDO squared. High leverage creates high prices: more buyers are willing to buy at a high price, because they can borrow – the marginal buyer can afford more when leverage is high. When people noticed that something was going wrong, margin (collateral) requirements increased, leverage decreased, price fell, worry rose, collateral requirements increased, etc.

Leverage poses many problems:

- A few investors, with high leverage, control the market (they are the marginal buyers) – what if they are not rational?
- Leverage spreads inequality.
- Leverage increases (leverages) volatility, which impacts economic activity.
- Some companies are deemed “too big to fail”.
- After a leverage crash, there is a debt overhang: no investments (from home owners and businesses) and no loans (from banks).

Preventing those problems is not that hard (in hindsight, it is never hard):

- For mortgages, just write down the principal of the loans (the alternative is to evict the home owners, but this is bad for everyone: people lose their home; banks only get back part of their money, after a few years, when the house is sold, at an abnormally low price (auction), even accounting for the drop in value because it was not maintained (or was vandalized) after the eviction). This is problematic because the bond owner and the home owner are not in contact – there is an intermediary, with no incentive to write down the principal.
- To prevent the leverage collapse, banks should lend with less collateral, e.g., by forbidding quick changes in the margin requirements.
- To replace the natural buyers, the government should invest more.
- To prevent or observe the next leverage crisis, one (the Federal Reserve) should not only monitor interest rates, but also the collateral.

Hyperbolic discounting is rational: valuing the far future with uncertain discount rates
J.D. Farmer and J. Geanakoplos (2009)

In a multi-period world, we want the discount factor for future utilities to be stationary and time-consistent.

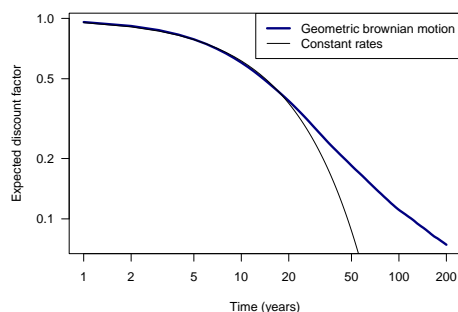
$$\text{Utility}_t(x) = \sum_{s \geq t} D_{t \rightarrow s} \text{Utility}(x_s)$$

With known, deterministic rates, the discount factor decreases exponentially. However, if the interest rate is stochastic, described by a geometric Brownian motion, then the expected discount factor exhibits a power law decay, at least after some time (a couple of generations).

$$D_{t \rightarrow s} = \prod_{\tau=t}^s (1 + r_{\tau})^{-1}$$

$$E[D_{t \rightarrow s}] \underset{s \rightarrow \infty}{\sim} (s - t)^{-\alpha}$$

There is no such effect for more realistic mean-reverting models.



Discussion of “the leverage cycle” H.S. Shin (2009) after J. Geanakoplos

Real-estate is a leveraged investment: home buyers using a mortgage with a 5% downpayment are leveraged 20 times. The leverage cycle can be modeled as follows. In a 1-period world, consider two assets, cash and stock, with the stock price distributed as $U(q - z, q + z)$ in period 1, and two types of investors, trying to maximize the same expected utility under different constraints: unleveraged investors have a leverage at most 1, leverage investors have a value-at-risk (VaR) constraint (since the price distribution is uniform and known, the constraint can be formulated as “they cannot be bankrupt”). When the asset price increases, the VaR constraint is less binding, non-leveraged investors can invest more, demand increases, and price increases even further – there is a similar downward spiral.

Demography and the long-run predictability of the stock market J. Geanakoplos et al. (2004)

Markets ups and downs, over the past century, seem to coincide with baby booms and busts: part of the effect can be explained by the equity premium, higher when the population is older. This can be modelled with an overlapping generations model, an incomplete market, and age-dependent utilities; the effects remains even after adding families (children helping parents), social security (idem), bequests (the opposite).

Game Theory B. Polak (Yale, 2007)

Game theory is the study of decision making, not when facing an unchanging world, as often in economics, but

when facing someone else, who can also take decisions that will affect your payoff.

There are many notions of a “good” (or bad) decision.

A **Pareto efficient** situation is a situation that cannot be changed to make one player better off without making the other worse off: since it usually requires cooperation between the players, it rarely happens.

A strategy is **dominated** by another strategy if it is always worse, whatever the other player does. (If you replace “worse” by “worse or as bad as”, that is a weakly dominated strategy.)

You can try to find a “good” decision, iteratively, by discarding the dominated strategies for both players, which gives a new game, and continuing to discard the dominated strategies until there are none left. However, this assumes that both players are rational, known that the other is rational, know that the other knows they are rational, etc. – *i.e.*, players are rational and this is **common knowledge**. In practice, it does not work.

A **Nash equilibrium** (NE) is a situation in which each player’s decision is a **best response** to the other player’s decision. Since the definition is circular, you can expect a few problems: it need not exist, need not be unique, need not be Pareto efficient, and weakly dominated strategies can be Nash equilibria (but strict ones cannot).

Randomized decisions can create (mixed) equilibria in situations with no pure equilibria, such as the rock-paper-scissors game. A pure strategy that is part of a mixed strategy NE is itself a best response (to the other participant’s strategy, which is usually a mixed strategy): mixed equilibria cannot be strict. To find a mixed strategy NE, look for mixes that make the other party indifferent. Mixed strategies can be interpreted as randomized strategies or as a population whose elements have different behaviours.

A strategy S is **evolutionary stable** if, when a new strategy appears, with a small proportion in the population, it disappears. Then, (S, S) is a Nash equilibrium. Conversely, if (S, S) is a strict Nash equilibrium, then S is evolutionary stable – this does not work with weak equilibria.

The payoffs of a simultaneous game can be represented as a matrix of (with two values in each cell, one for the row player, one for the column player). The decisions and payoffs in a **sequential game** can be represented as a tree. **Backward induction** gives a “good” solution to the game, but it assumes that the other player is rational – and it need not be Pareto efficient: there can be **moral hazards**.

Backward induction assumes that the other player will play it his/her best interest, *i.e.*, that they will not choose something that would be bad for them and for us. It assumes they will not “screw up”. It does not give a worst-case strategy.

You can tweak a simultaneous game by changing the payoffs (**incentive design**: give some of your payoff to the other player, in some situations, to encourage him/her to choose that path – you will get a smaller share of a larger pie), or pruning the tree of unwanted branches by restricting your choices (build a new factory, which becomes a sunk cost; burn your ships to show your **commitment**, etc.).

Zermelo's theorem says that in a sequential game with perfect information and a finite number of moves, either player 1 can force a win, or player 1 can force a tie, or player 2 can force a win.

With imperfect information, many Nash Equilibria are clearly bad choices. A Nash Equilibrium is a **subgame perfect equilibrium** (SPE) if it also induces a NE on all subgames.

Repeated games can lead to **wars of attrition**: past costs are sunk, they do not play any role in new decisions, and can accumulate, forever. This happens in wars (costly wars over very small frontier areas), companies fighting for a market regardless of the cost or bribery auctions ("all pay auction": two companies bribe a government until one out-bribes the other).

Repeated games try to enforce cooperation by balancing threat of future punishments and prospects of future rewards. If there is a time limit, they can exhibit the **lame duck** effect (politicians or CEOs approaching retirement): in the last period, you no longer need to cooperate, because there is no future – and, by backward induction, there is no reason to cooperate at the beginning either. In some cases, it can work: if there are several Nash equilibria, you can use one as a reward and the other as a punishment, to encourage cooperation.

This effect can disappear in repeated games with a random number of periods (e.g., in the prisoner's dilemma, one can use the grim trigger strategy: cooperate until the other defects – to show that it is an equilibrium, discount the future using the probability that the game will continue).

When information is asymmetric, some players may want to reveal the information while other will try to hide it – but the information has to be verifiable: claiming "our product is the best" is not sufficient, even if it happens to be true. Not revealing information is informative: it means that it is not in your interest to do so (information unravelling: "Silence speaks volume").

Revealing verifiable information can be costly. For instance, in a world with two types of workers, good and bad, and two types of pay, high and low, employers and good workers want the distinction to be known, bad workers do not. If the cost of education is sufficiently different for good and bad workers, good workers will spend time to earn a degree, but bad workers will not: this is a **separating equilibrium**. This can lead to qualification inflation: if the cost difference is too small, good workers will try to earn even more degrees. (This model assumes that education does not

teach you anything, or at least nothing useful for your future job: it is socially wasteful, uses resources but does not improve productivity)

Auctions are becoming more important (governments selling oligopoly rights (phone markets, natural resources), ebay, IPOs, microstructure of stock markets, etc.). In a common value auction, the value is the same for everyone, but it is unknown. In a private value auction, the values are different for everyone and unrelated. (Reality is between those two extremes: an oil well has a common value, but different companies may have different exploitation costs; a house has a private value (consumption), but since you eventually resell it, other people's values matter.)

In a common value auction, if everyone bids their (unbiased) estimated value, the winner will overpay: this is the **winner's curse**. If you win, it is bad news. (This often happens with IPOs.) You should bet under the assumption that you will win, *i.e.*, you should ask yourself "What is your estimate? You have won, so your estimate is too high: do you want to correct your estimate?"

There are many types of auctions: first price sealed bid auction; second price sealed bid auction (or victory auction: the highest bidder gets the good, but pays the second highest bid); ascending open auction (the price progressively rises, and people progressively drop out – very similar to the second price sealed bid auction); descending open auction (or Dutch auction: the price progressively falls until someone accepts it – identical to the first price sealed bid auction). The optimal strategies are different: in a private value second price sealed bid, you should bid your value: it is a weakly dominant strategy; in a private value first price sealed bid auction, bidding your value is weakly dominated: you should bid slightly less. Surprisingly, the expected revenue generated by these auctions is the same.

Here are some of the games or real-world situations used to illustrate those notions.

Coordination games are variants of the prisoner's dilemma: competing or colluding firms, overfishing, global warming (contractual enforcement is needed: it is not just a communication problem), etc.

In the number game, each of n participants chooses a number between 1 and 100, the winner is the one closest to $2/3$ of the average (this shows that, in practice, if you delete dominated strategies until the end, you lose: rationality is not common knowledge).

Alice and Bob both wear a pink hat but do not know the colour of their hat: "Someone is wearing a pink hat" is shared knowledge, but not common knowledge.

In the election game (linear city model), two politicians decide on their position on the political spectrum $[1, 10]$; each position has the favour of 10% of the population; people vote for the closest candidate (or one of the closest candidates, at random, in case of a tie): iterative deletion of dominated strategies leads to both

candidates choosing a medium position (5 or 6). This also applies to product placement or petrol station location (the best location is next to a competitor, not in an area with no competition).

In the investment game, each of N players can invest \$0 (no loss, no profit) or \$10; if more than 90% choose to invest, they receive \$5 more, if not, they lose their \$10. There are two Nash equilibria: everyone invests, or no one does. It is a cooperation game, but contrary to the prisoner's dilemma, there is no dominated strategy: contractual enforcement is not needed, communication or leadership often suffices to move to the better Nash equilibrium. Bank runs, monopolies (the use of Microsoft products as a de facto standard) or fashion trends are similar examples.

The **Cournot duopoly** (two firms competing on the quantity produced) is a game of **strategic substitute**, the opposite of a coordination game: if you produce more, the other party produces less (in a coordination game, if you produce more, the other party has an incentive to produce more). The Nash Equilibrium does not maximize total profit; monopoly or collusion does, but collusion is not a Nash equilibrium: the firm would have an incentive to produce more.

In **Bertrand competition**, the two firms compete on prices, not quantities: the outcome is similar to perfect competition.

In the candidate-voter model, each voter is a potential candidate, and their political opinion is in $[0, 1]$; there is a reward for the winner, a cost to run, and a cost proportional to the square of the distance between the voter and the elected candidate. There are many Nash equilibria: a single candidate exactly in the middle, two candidates symmetrically positioned around the centre (but not too far from the centre, otherwise a centrist candidate could run and win). This mimics what happens in real elections: if a new candidate appears on one side, the other side wins.

In the location model, two types of people have to choose to live in one of two cities; they prefer a heterogeneous environment, but want to avoid being isolated among too many people of the other type. The heterogeneous situation is better for everyone, but it is not stable.

Consider n lions, ranked by seniority, and one sheep. Only lion 1 can eat the sheep, but lion 2 will then have the opportunity to eat him in his post-meal nap, and so on for the other lions. Should lion 1 eat the sheep? (Use backward induction and consider the behaviour of the last lion.)

In a Cournot duopoly, if firm 2 has a spy that tells them what firm 1 will be doing, and firm 1 knows it, then firm 1 has a **first mover's advantage**. For firm 2, more information, more choices can hurt.

In the game of Nim, there are two piles of stones. You can remove as many stones as you want from either pile, but if you take the last stone, you lose. (The strat-

egy is to make the two piles equal.) There is a second mover's advantage if the two piles are equal, but a first mover's advantage otherwise. Here is a 2-dimensional variant: the stones are in a $n \times m$ array; choose a stone and remove it and all the stones north-east of it; you lose if you remove the last stone.

Two duellists are walking towards each other; if their skills (probability of hitting the adversary as a function of the distance) are known, when should they shoot? (Not too early (if $p_n < 1 - p_{n+1}$, shooting is a dominated strategy), not too late (use backwards induction to show that you should act as soon as $p_n \geq 1 - p_{n+1}$).) A similar situation appears in economics, when two companies are developing a new product, want to release it before the other to occupy the market, but do not want to release it too early because it is not completely finished.

The **chain store paradox** shows the advantage of not being rational (or, at least, having the other party believe you are not): several firms consider entering a new market, currently controlled by a monopolist; if it is more costly for the monopolist to fight the new entrants than to leave them a share of the market, the monopolist should not fight – but if they have the reputation of being irrational (systematically or randomly fighting, even if it is not profitable), they can deter new entrants.

The **dictator game** (you are given \$1, you can give part of it to the other player, who cannot do anything) and the **ultimatum game** (you are given \$1, you can give part of it to the other player, who can accept or reject the offer, in which case no one receives anything). show that we are not always rational. Bargaining is similar to the ultimatum game, but with several periods, and a discount factor (the value to share decreases with time). The first offer is always accepted; the first mover's advantage disappears as the number of periods increases; if discounting is almost negligible, and similar (and known) for both players, an even split is optimal. However, if the value for each player is not known, the result of the bargaining is suboptimal.

When designing incentives, do not overlook the strategic effects. Consider two firms competing in a Cournot equilibrium. One has the opportunity to invest in new equipment: it is expensive, but will reduce production costs. If the quantity produced does not change, it may not be profitable, but if the other firm knows that the investment has been made, there is a strategic effect: they will reduce their production, we will increase ours, and the new investment will become profitable. Similar effects appear in the tax code.

Comparison of correlation analysis techniques for irregularly-sampled time series K. Rehfeld et al. (2011)

The auto-correlation and cross-correlation of irregularly-sampled time series can be estimated by considering the time series as regular; linearly in-

terpolating to have regular time series; binning the observations pairs by lag, *i.e.*, considering a regression

$$(x_t - \bar{x})(x_s - \bar{x}) \sim |t - s|$$

with a rectangular kernel; or using a similar regression with a gaussian (this is what gives the best results) or a sinc kernel.

***A drunk and her dog:
an illustration of cointegration
and error correction***
M.P. Murray (1994)

A set of random walks

$$x_n - x_{n-1} = u_n$$

$$y_n - y_{n-1} = v_n$$

can be turned into cointegrated random walks by adding error correcting terms

$$x_n - x_{n-1} = u_n$$

$$y_n - y_{n-1} = v_n + a(y_{n-1} - x_{n-1}).$$

When looking for cointegration relations in data, make sure they are meaningful (here, $x_{n-1} - y_{n-1}$ is the distance between the drunk and her dog).

***A drunk, her dog and a boyfriend:
an illustration of multiple cointegration
and error correction***
A. Smith and R. Harrison

The model can be generalized by adding a boyfriend

$$x_n - x_{n-1} = u_n$$

$$y_n - y_{n-1} = v_n + a(y_{n-1} - x_{n-1})$$

$$z_n - z_{n-1} = w_n + b(z_{n-1} - x_{n-1}),$$

the drunk's attraction for her dog or boyfriend, the boyfriend's aversion for the dog, etc.

By looking at the data alone, you cannot always answer the question "who is attracted by whom?": you only have the subspace of cointegration relations. If the cointegration relations are $\beta' \mathbf{x}_n$ (this means that this series is stationary), the error correction model (ECM) is $\mathbf{x}_n - \mathbf{x}_{n-1} = \alpha \beta' \mathbf{x}_{n-1} + \mathbf{u}_n$; we can estimate $\alpha \beta'$ (and its rank, *i.e.*, the number of independent cointegration relations), but not α and β separately.

***Nonparametric goodness-of-fit tests
for discrete null distributions***
T.B. Arnold and J.W. Emerson
R Journal (2011)

The Kolmogorov-Smirnov and Cramér von Mises tests use the L^∞ or L^2 (or generalizations) distances between the cumulative distribution functions as test statistics: for continuous distributions, the distribution of the test statistic does not depend on the hypothesized distribution. For discrete distribution it does: the **dgof** package can compute the corresponding p -values.

***Ckmeans.1d.dp: optimal k-means clustering
in one dimension by dynamic programming***

In one dimension, the (usually NP-hard) k -means problem is amenable to dynamic programming (consider $D_{i,m}$, the minimum within sum of squares when clustering x_1, \dots, x_i into m clusters) – for situations where reproducibility is important.

***Tweets and peers: defining industry groups
and strategic peers based on investor
perceptions of stocks on Twitter***
T.O. Sprenger and I.M. Welpe (2011)

Traders tag their company-related messages with the ticker (e.g., \$AAPL, which gets indexed as @MV_AAPL): company co-occurrence can be used to define industry groups; the classification reflects structural changes more quickly than traditional methods (looking at the variance matrix of the returns, looking at analysts in common, etc.).

Zipf's law unzipped
S.K. Baek et al. (2011)

The omnipresence of Zipf's law (distribution of city sizes, word frequencies, etc.) can be explained using information theory: when putting M balls into N boxes, find $N(k)$, the number of boxes of size k to maximize the entropy (this can be reformulated in terms of mutual entropy). Contrary to the law of large numbers, it is not a process of aggregation, but of division.

***Stability of the world trade web
over time – an extinction analysis***
N. Foti et al. (2011)

Yet another study of the world trade web (WTW) dataset, modelling what happens to the trade network if one or more nodes or edges are removed or modified.

***Full characterization
of the fractional Poisson process***
M. Politi et al. (2011)

You can build counting processes (random variables $N(A)$ that count the number of points in a Borel set A) by specifying the distribution of the inter-event times: with exponential inter-event times, this is a Poisson process (Markov, *i.e.*, memory-less), but other power-law distributions, e.g., Mittag-Leffler, give a non-Markov process.

***Price dynamics
in a Markovian limit order market***
R. Cont and A. de Larrard (2011)

Most models of the limit order book are only amenable to simulations; by considering a simplified model, with the level-1 order book filled or depleted by Poisson processes for market, limit or cancellation orders, unit order sizes, unit spread, and a prescribed order book distribution after a tick changes, one can compute

many quantities of interest: distribution of the duration between price changes, autocorrelation of the price changes, volatility, etc.

A stochastic model for order book dynamics
R. Cont et al. (2010)

A more complicated (continuous time) model.

**Option pricing and estimation
of financial models with R**
S.M. Iacus (Wiley, 2011)

This book (and the accompanying `sde`, `yuima`, `opefimor` packages) explains how to estimate continuous-time models with R and use them to price options. The theoretical chapters also contain *exercises* to ensure that the reader understands the notions introduced.

R already provides `dpqr` (density, probability (cumulative density function, cdf), quantile (inverse of the cdf) and random sample) functions for standard distributions; the `fBasics` package adds a few more: `nig` (normal inverse gaussian, *i.e.*, hitting time of a random walk), `gh` (generalized hyperbolic), `stable`.

Some of those distributions are only defined from their characteristic function (this is the case for infinitely divisible distributions in general: these are the distributions whose characteristic function is given by the Lévy-Khintchine formula). The fast Fourier transform (`fft`) can be used to move between cumulative distribution function F (cdf) or the probability density function f (pdf) and characteristic function ϕ .

$$\begin{aligned} F(x) &= \frac{1}{2} - \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{e^{-itx}\phi(t)}{it} dt \\ &= F(0) - \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{e^{-itx} - 1}{it} \phi(t) dt \\ f(x) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-itx} \phi(t) dt \end{aligned}$$

The `mle` function can be used to maximize a log-likelihood: the result itself is the same you would get with a general-purpose optimizer (`optim`), but you can call functions such as `summary`, `vcov`, `confint` on the result to have more information (`fidistrplus` provides more, moment-based, estimators, and allows censored data).

$$\begin{aligned} \text{Score} &= \frac{\partial \text{LogLik}}{\partial \theta} \\ E[\text{Score}] &= 0 \\ \text{Information} &= \text{Var} \frac{\partial \text{LogLik}}{\partial \theta} \\ &= -E \frac{\partial^2 \text{LogLik}}{\partial \theta^2} \\ \text{Var}(\text{Unbiased estimator}) &\geq \frac{1}{\text{Information}} \end{aligned}$$

Telegraph processes (random motion with constant velocity $\pm c$, and direction changes given by a Poisson process) lend themselves to explicit computations (but beware: they have finite variation and allow for arbitrage opportunities). Their limit, as the Poisson density increases, is a random walk.

Lévy processes can be defined as (limits of) the sum of a Brownian motion with drift and (several) compound compensated Poisson processes. Alternatively, they are **stochastically continuous** (intuitively: there can be jumps, but they are random, and if there are infinitely jumps on a compact (infinite activity), they are not too large) processes with stationary independent increments. They are characterized by their characteristic function. Ito's formula can be generalized to Lévy processes.

The `numDeriv` package can compute derivatives, increasing the precision by evaluating the function at more carefully chosen points (e.g., $t+h$ and $t-h$ instead of t and $t+h$) or with more general methods (Richardson interpolation).

Non-homogeneous Poisson processes can be simulated with the acceptance-rejection method ("thinning"): sample events with constant intensity $\lambda \geq \lambda(t)$; for each event, take a random number in $[0, \lambda]$; if the number is not in $[0, \lambda(t)]$, reject the event.

The `sde::sde.sim` function can sample from the solution of a stochastic differential equation, using the Euler or Milstein scheme (based on first or second order Taylor expansion); the `yuima` package generalizes this to multi-dimensional processes, Markov switching diffusions and Lévy processes (a Lévy process is a time-changed Brownian motion).

Stochastic models (the drift and sensitivity of a diffusion) can be estimated via maximum likelihood if the transition probability is known (normal, log-normal, CIR, etc.), or via **quasi-maximum likelihood**, *i.e.*, by replacing the continuous stochastic differential equation with its first order discretization (implementation in the `sde` and `yuima` packages).

Interest rates are often modelled as

$$dX_t = f(X_t)dt + \sqrt{g(X_t)}dB_t$$

where f and g are (Laurent) polynomials (of given degrees and unknown coefficients – or arbitrary functions). They can be fitted (with bias) via 2-stage regression:

$$\begin{aligned} X_{t+1} - X_t &\sim f(X_t) \\ \text{res}_{t+1}^2 &\sim g(X_t) \\ X_{t+1} - X_t &\sim f(X_t), \text{ with weights } w = g(X_t) \end{aligned}$$

The `fBasics` package provides many functions to fit non-gaussian distributions (`nigFit`, `hypFit`, `ghFit`, `stableFit`).

The Black-Scholes PDE (partial differential equation), for the price of a European option, is just a consequence of Ito's formula, for a self-financing portfolio.

More generally, since the Gaussian density is a solution of the heat equation, one can show that (functions of) expectations of functions of a random walk are also solution of some PDE.

The `fOptions::GBSOption` function computes Black-Scholes option prices.

The equivalent martingale measure can be computed in as follows.

$$\lambda = \frac{\mu - r}{\sigma}$$

$$M_t = \exp(-\lambda B_t - \frac{1}{2}\lambda^2 t)$$

$$Q(A) = E[\mathbf{1}_A M_T] \text{ (martingale measure)}$$

$$W_t = B_t + \lambda t \text{ brownian motion under } Q$$

Monte Carlo option pricing can be sped up by parallelizing the computations, e.g., with the `foreach` and `doSnow` or `doMC` packages.

```
library(doMC) # multicore
library(foreach)
registerDoMC(4) # 4 cores
p <- foreach(m=1:4) %dopar% f(m)
```

Now that the `parallel` package is a core package, the following suffices.

```
library(parallel)
r <- mclapply(
  1:1e5, f,
  mc.cores=detectCores()
)
```

Option sensitivities (“greeks”), *i.e.*, partial derivatives of option prices can be computed in many ways:

- Using simple formulas, but they often involve the derivative of the payoff, which is rarely differentiable;
- Using the same formulas, after integration by parts, to get rid of the derivative;
- By numerical approximation of $\partial C / \partial S$;
- By Monte Carlo simulation (the sensitivities can be expressed directly as integrals, without having to compute the option price);
- Monte Carlo simulation for $C(S, t)$ and $C(S + h, t)$ (using the same random seed for both to improve precision).

Asian options can be priced via an asymptotic expansion $dX_t^\varepsilon = a(X_t^\varepsilon, \varepsilon)dt + b(X_t^\varepsilon, \varepsilon)dW_t$ around a deterministic process using Malliavin calculus (?).

Options on baskets are priced in exactly the same way.

`fOptions::GBSCharacteristics` computes sensitivities, `fExoticOptions` and `fAsianOptions` implement many pricing algorithms, `GBSVolatility` computes the implied volatility, `sde::cpoint` looks for structural breaks (in the volatility of a diffusion).

American options can be priced via dynamic programming, using the grid of the finite difference method,

or using regressions to move back in time (Longstaff-Schwartz least squares method). The Monte Carlo estimator is biased, but one can easily compute an upper and a lower bound. There are also various approximations, e.g. with a sequence of Bermudan options (which can only be exercised at specific times) and Richardson’s extrapolation, or by finding an ODE approximately satisfied by the early exercise premium *i.e.*, the difference between the prices of the American and European calls (quadratic approximation).

If the price is an (exponential) Lévy process, the equivalent martingale no longer unique. Upper and lower bounds for the no-arbitrage price are known, but the interval is very large. One can try to transform the density f of Z_1 in some way, so that the density remains infinitely divisible, but turns the process into a martingale; for instance, an **Esscher transform**,

$$f_\theta(x) \propto e^{\theta x} f(x),$$

or a mean-corrected martingale measure,

$$\tilde{Z}_t = Z_t + mt.$$

The prices of European calls and puts can be expressed as integrals involving the characteristic function (of the real-world distribution) of the price at expiry, and can be computed via numeric integration or FFT.

The **benchmark approach** suggests to stay with the physical measure, but to devise a discount factor that discounts both time and risk, *i.e.*, so that the discounted price be a martingale. This stochastic discount factor (or deflator) corresponds to an asset, called a benchmark, used as a numéraire: if prices are expressed in terms of the benchmark, they are martingales. The growth-optimal portfolio is a benchmark.

The book ends with a discussion of change point detection, estimation of the covariance of asynchronously observed diffusions (Hayashi-Yoshida estimator), regularized estimators (lasso) and clustering with the Markov operator distance.

UseR! 2011

(I did not attend the conference but just read the presentations online: they were only made available six months after the conference, and only half the talks had a PDF file.)

The results of most statistical analyses are known to be “asymptotically” correct: they are only valid if you have enough data (estimators are asymptotically gaussian, the p -values displayed are asymptotically uniform under the null hypothesis, etc.). For small samples, the results are not only imprecise because there is not enough data to reduce the statistical noise, they are also biased. It is often possible to reduce this bias, with **higher order asymptotic expansions** (of the distribution of the estimators or test statistics). The `pbrktest` package does that for mixed models; the `metaLik` package does that for mixed models used for meta-analyses (*i.e.*, to aggregate data from different

studies); the **brglm** package does that for binomial GLM (“binomial” means that we are predicting proportions).

Sweave is no longer the only way of generating PDF reports from R:

- The **brew** package uses templates (not unlike Php) to generate HTML, L^AT_EX, or any other kind of file;
- The Emacs org-mode, with babel, not only helps you organize your notes (if you like to put everything you do in a single text file, that grows by 1MB every year – it is more powerful than supposedly modern note-taking applications such as Tomboy), it can also include bits of code, in any language (elisp, shell, Python, R, C, etc.), that are executed and whose results are displayed (as text, as a table, as plots), and can generate PDF, HTML, etc.
- **knitr** is a replacement for Sweave.
- The **tikzDevice** package generates plots using pgf commands, suitable for inclusion in L^AT_EX files: one advantage is the consistent use of fonts and font sizes between the text and the plots.
- Tables can be included in your L^AT_EX reports thanks to the **xtable**, **tables**, **Hmisc** (and the **latex** and **describe** commands), or **compareGroups** (for the complicated tables generated by **ddply**) packages.
- The **sparkTable** package can add sparklines to your tables.

Interactive plots remain one of R’s weaknesses, but

- the **animator** package can generate animations (e.g., to illustrate random phenomena in introductory statistics courses);
- the **googleVis** package can generate interactive plots in JavaScript or Flash, similar to the animated bubble plots popularized by H. Rosling.
- It is possible to create interactive SVG documents (that can be displayed in a web browser, or included in a web page), e.g., with the **gridSVG** package (the example also included the conversion of a PDF file (a map) to a vector format, for further processing, with **grImport**).
- The interactive graphics available with R (**iplots**, **rgl**, **rggobi**, **playwith**) still pale when compared with Javascript (D3, jit, Raphaël, processing) or even Python.

Since the development of R packages is not centralized, we often end up with several packages doing similar but complementary things, with completely different interfaces. There is some continued work to compare, standardize, refactor those implementations, but the road will be very long...

- The **betareg** package, implementing beta regression (used to model quantities in the interval (0,1): one could also use logit or probit regression, but that would model $\text{logit}(y)$, which has no interpretation – we can directly model the quantity of interest instead) can leverage the **flexmix** package for latent class (mixture) beta regression (in the **betamix** function) and the **party** package for beta regression trees

(**betatree** function).

- The **ordinal** package provides a unified interface to the various packages dealing with ordinal regression (aka ordered probit/logit): **MASS::polr**, **Design::lrm**, **VGAM::cumulative**, **MCCglmm**.
- The **binomTools** package does the same for binomial regression (predict a proportion), using some of the functions in **boot**, **faraway**, **car**, **MLDS**.
- The theme of the plots generated via base graphics, **ggplot2** and **lattice** is completely different: the **uniPlot** package can uniformize them, in case you mix them in the same document.
- The **ROI** and **optimix** packages (both) try to provide a single interface to the various optimization (mathematical programming) algorithms available.
- R, Jags and ADMB have been compared on a few dozen sample problems.
- There are no fewer than 25 density estimation functions in R: a comparison of their speed and precision suggests to use **KernSmooth** or **ASH**.

If you use R more as a programming language than a statistical environment:

- There are no fewer than three packages for unit tests: **testthat** (if you hesitate, use this one), **RUnit** and **svUnit**.
- The **ROpenCL** package (inspired by PyOpenCL) lets you program your GPU, and may supercede **gputools**, **rgpu** (obsolete), or **cudaBayesreg** (which are mostly NVidia-specific).
- CXXR is a reimplement of R in C++, in case you need to tweak R’s internals, e.g., to add new types (big integer, etc.) – could it be used to replace most R structures with out-of-memory data, in a way completely transparent to the end user?
- There are no fewer than three object-oriented paradigms in R: S3 (function-centric), S4 (class-centric) and R5 (mutable objects, à la Java, aka reference classes). The need for specific packages (**R.oo**, **proto**, **mutatr**, etc.) may disappear.
- The RStudio IDE is getting more and more popular: it may have already replaced Eclipse/StatET (which I never managed to install properly: too Javaesque). But Emacs/ESS is still there.
- R Commander still serves as a base to build domain-specific GUIs (for users who do not want to, or think they cannot, use R, but eventually will), but the **gwidgets** package (for real GUIs, for users who will not use R) is getting more widely used.
- R is now available “in the cloud”, mostly through commercial companies (some small companies, targeting niche markets, start to compete with giants like Amazon), but many universities and research organizations do the same (for instance, Rc2 is used to allow students to access R in a workbook-like fashion).
- Commercial support for R is available through companies such as Revolution Analytics, who also provides a few parallel external-memory algorithms and a GUI.
- It is now easy to perform parallel computations in R,

with the `parallel`, `foreach`, `multicore` and `snow` packages – but it is not magical: you still have to split the data or computations and merge the results yourself.

There are many, many types of regression models: the relation can be linear, non-linear (with a given functional form) or non-parametric (with no given functional form); we can try to directly predict the quantity of interest (Gaussian model), or some parameter of the model (GLM), or some quantity that depends on the model (e.g., quantile regression) or the whole distribution of those quantities (Bayesian statistics); the cost function can be a sum of squares, a robust cost function, or include a penalty term (regularized regression, e.g., ridge (L^2), lasso (L^1) or elastic net (both)); the observations can be weighted, or even dependent: the shape of the correlation matrix is usually known, and can correspond to temporal or spacial correlation, or to groups in the data; the model can have several levels, as in “for each subject, take the model parameters at random following some probability distribution, then take 10 samples from those models for each subject, and try to recover the initial probability distribution” – this is actually equivalent to imposing a correlation structure on the data; the model can have “nuisance” parameters, *i.e.*, parameters whose value we are not interested in (e.g., subject-specific parameters in the previous example, or the hidden states in a hidden Markov model, etc.).

If you just want to analyze data, you expect to be able to mix and match those blocks at will – unfortunately, it does not work like that, not only because those methods were developed independently, by statisticians (not programmers), as proofs-of-concept, for a particular application, but also because the algorithms are complicated, time-consuming, and often incompatible. Some progress is being made, not really in uniformizing everything, but in filling in the missing combinations: `lmer`, `glm`, `glmnet`, `glmer`, `glmmlasso` (penalized mixed GLM), `lqmm` (quantile regression and mixed models), etc.

I always complain that copulas are useless (or, at least, unused and difficult to use) in high dimensions, where dependency is trickier to estimate. **Gaussian Copula marginal regression** (in the `gcmr` package) is such an example. You can model a time series y_i as

$$y = F_{\theta}^{-1}(\varepsilon_i)$$

where F is the (marginal) distribution, from some family of distributions (e.g., Gaussian with mean and variance to determine – this is where you would add predictive variables, e.g., $N(a + bx, \sigma)$, with a , b , σ to determine) and the innovations

$$\varepsilon \sim N(0, \Omega)$$

are multivariate normal, with Ω describing the dependency structure, e.g., ARMA for a time series (as with GLS or mixed models). The model can be fit via maximum likelihood (there are complications for discrete data).

The `missMDA` package implements **iterative PCA**, *i.e.*, PCA (principal component analysis) with missing values, imputed via expectation maximization (EM). Multiple imputation generates several values for each missing observation (otherwise the data does not look as variable as it should): in the PCA plots (check the `FactoMineR` package), observations (or variables) are replaced by ellipses or clouds or points. The same ideas can be applied to qualitative variables (multiple correspondence analysis (MCA)).

A qualitative variable with n possible values can be described with a multinomial model, which requires $n - 1$ parameters (the probabilities). The model can be simplified (multinomial processing tree model) by describing the process leading to the outcome with a tree, whose nodes are Bernoulli trials (whose probabilities have to be determined) and whose leaves are the possible values. (I am puzzled: the number of parameters is the same, unless you assume that some of those parameters are equal – it is not really a simplification, just a rearrangement of the model parameters.)

To find the value of a parameter θ of a model $M(\theta)$ describing your data, **Approximate Bayesian Computation** (ABC) suggests to generate random data from $M(\theta)$, for various values of θ , and compare with your data, using some summary statistics. This is just a parametric-bootstrap-based generalized method of moments (GMM). The `ABCME` package chooses the moments itself by minimizing entropy.

The `ClustOfVar` package clusters variables with PCAMIX (as you cluster observations, in PCA, etc.), even if you have both continuous and discrete variables.

Emulation is the process of replacing complicated, time-consuming (deterministic) functions of many variables with a statistical approximation; it has been used in meteorology for some time now, and starts to spread to other domains.

There is a lot of activity around social media mining: data can be retrieved from the web with the `RCurl`, `XML` or `RJSONIO` packages, cleaned with `mvoutlier`, the missing data can be imputed with `mice`, modelled with regression trees with the `BayesTree` or `party` packages. Bayesian networks are sometimes used, through the `bnlearn` and `pcalg` packages.

Examples in various domains were presented, such as finance (the `robHMM` package, for robust HMM or Markov switching models, can be applied to value vs growth portfolio selection or option pricing, with two volatility regimes accounting for kurtosis and skewness), astronomy (galaxy positions, from the hyperLEDA database), business (e.g., how to manage inventory to avoid or limit the bullwhip effect: if demand fluctuates, the fluctuations are amplified along the supply chain because of the lack of visibility of the end-user actual demand), quality control (six sigmas) and metrology, biology (as usual), teaching, geographical information systems (GIS), etc.

Nomograms for visualising relationships between three variables

J. Rougier and K. Milner
(UseR! 2011)

Nomograms are an old-fashioned way of representing linear or non-linear relations between three variables, as in a $y \sim x_1 + x_2$ regression. It is sometimes easier to add more information (another response variable, differences between several groups, etc.) to a nomogram than to the more modern-looking levelplot.

Gaussian copula marginal regression

G. Masarotto and C. Varin

More details.

Implementing models in quantitative finance: methods and cases

G. Fusai and A. Roncoroni
(Springer, 2008)

The first part of the book presents algorithms you may need or find useful when implementing quantitative finance models, concisely, with non-trivial examples.

Antithetic variables can reduce the variance of Monte Carlo estimators: if X and Y have the same distribution, then $E[f(X)] = \frac{1}{2}(EfX + EfY)$, but if $\text{Cor}(fX, fY) < 0$, the variance of the corresponding estimator is lower. This is often used with $Y = -X$, to ensure that the sample points symmetrically spread out.

Control variables use a similar idea: notice that $EfX = E[fX - \alpha(gX - EgX)]$, and choose a function g close to f , but for which you can compute EgX , and select α to minimize the variance of the corresponding estimator. This can be used to price an Asian option with arithmetic average (f , no closed formula) from the price of an Asian option with geometric average (g , closed formula available). This can also be used to price options in a non-Black-Scholes model (f) using the Black-Scholes prices, *i.e.*, the value of the Δ -hedged strategy (g).

Importance sampling can be used to accurately price out-of-the-money options, by sampling from the exercise region. Option pricing usually uses drift-less processes (the risk-free measure): with importance sampling, you can actually add a drift, so that the process visits the exercise region more often, and select it to minimize the variance of the estimator.

Diffusion processes can be sampled exactly (if they can be expressed as a function of a brownian motion, $X_t = f(W_t)$, you just have to sample from a brownian motion), using the exact transition distribution $X_{t+\varepsilon}|X_t$ (if it is known), or by discretizing the stochastic differential equation: the Euler method uses a first-order approximation with gaussian innovations (you could use Bernoulli innovations: the convergence is only in law, and not pathwise, but this is sufficient to compute expectations); the **Millstein scheme** uses a second-order approximation.

Gaussian processes can be sampled for various series expansions: the Karhounen-Loeve expansion (complex sine function, that come from the diagonalization of the covariance operator, $\text{Cov}(X_t, X_s) = \text{Min}(t, s)$), the Haar function expansion ($\neg\bigwedge = \int \neg\bigwedge_{\neg}$) or the Paley-Wiener expansion (with real sine functions).

To simulate Poisson processes, you can simulate the inter-event times (exponential) until you reach the end of the desired period (countdown simulation); or first sample the number of events in the period: (since Poisson processes are memory-less) the event times are then independent and uniformly distributed (conditional simulation). This can be generalized to jump processes: if the jump intensity is variable, or even random, first sample the event times from a Poisson process with a higher jump intensity, then reject some of the events.

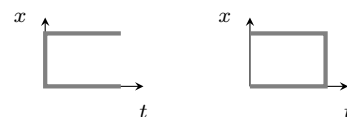
Stochastic dynamic programming can be used to price American options

$$\text{Price} = \sup_{\tau \text{ stopping time}} E^{\text{risk-free}}[e^{r\tau} \times \text{Payoff}(S_{\tau})].$$

There are several possible discretizations of the problem to a linear lattice or tree: you can choose the up and down jump sizes and their probabilities to give a correct approximation of $E[\Delta x]$ (first order) or both $E[\Delta x]$ and $E[\Delta x^2]$ (second order) – since there are four parameters, you can still impose equal jump sizes, so that the binomial tree reduces to a lattice.

Using stochastic dynamic programming for optimal investment requires the estimation of $V(t_k, w, s)$, the maximum value attainable in $[t_k, T]$, starting with wealth w and asset prices s , for all k – these are functions of $n + 1$ variables (w, s) .

The partial differential equations (PDE) occurring in finance are parabolic, *i.e.*, there is only one second order term, of the form ∂_{xx} , as in the heat equation (other types of PDEs include the elliptic type, like the Laplace equation, defining harmonic functions, and the hyperbolic type, like the wave equation, $\partial_{tt} - \partial_{xx}$ or $\partial_{tt} - \Delta$). To determine a solution, you need to impose some initial (or final) and boundary conditions, involving the function itself (Dirichlet condition) or its derivative (Neumann condition) or both (Robin condition).



For option pricing, the payoff is the initial condition, and the boundary conditions come from the asymptotic option price or the presence of a barrier (e.g., knock-out barrier options). (Discrete monitoring of barrier options was mentioned, but the boundary conditions did not look sufficient to me: there were too many holes in the boundary to uniquely define a solution.)

By a change of variables, the Black-Scholes equation can be reduced to the heat equation. For the heat

equation, the three approximations of the derivative,

$$\frac{f(t+h) - f(t)}{h}, \quad \frac{f(t) - f(t-h)}{h}, \quad \frac{f(t+h) - f(t-h)}{2h},$$

(forward, backward, central) lead to three algorithms, explicit (the computation of $f(t+h, \cdot)$ from $f(t, \cdot)$ is just a recurrence relation), implicit (we have to solve a tridiagonal linear system) and Crank-Nicolson (idem), of orders $O(h)$, $O(h)$, $O(h^2)$. Those algorithms are sensitive in discontinuities in the initial condition (option payoffs are rarely \mathcal{C}^1) and high values of $\alpha = \delta t / \delta z^2$ (the volatility always appears as $\alpha \sigma^2$).

Tridiagonal systems can be solved via the *LU* decomposition (it is bi-diagonal), *i.e.*, Gaussian elimination, or via iterative methods, $x_{n+1} = f(x_n)$, after rewriting the equation as $x = f(x)$, which progressively increase the precision of the result. For the Jacobi method, write $A = D + L + U$ (diagonal, lower and upper triangular) and $Ax = b$ becomes $x = -D^{-1}(L+U)x + D^{-1}b$. The **Gauss-Seidel** method uses the same formula, but with $x_{n+1,i}$ instead of $x_{n,i}$ if it has already been computed: $x_{n+1} = -D^{-1}Lx_{n+1} - D^{-1}Ux_n + D^{-1}b$. The Gauss-Seidel method can also be written $x_{n+1} = x_n - (D + L)^{-1}(Ax_n - b)$, where $Ax_n - b$ is the current error, used to correct the current estimate. The **SOR** (successive over-relaxation) method corrects a little more, $x_{n+1} = x_n - \omega(D+L)^{-1}(Ax_n - b)$, with $\omega \in]1, 2[$. There are also non-stationary iterative methods ($x_{n+1} = f_n(x_n)$, *i.e.*, the function changes each time), such as the conjugate gradient (it assumes that the matrix A is symmetric: if not, solve $A'Ax = A'b$ instead). In all cases, the speed of convergence depends on the eigenvalues of iteration matrix.

To numerically compute an integral on a small interval $[a, b]$, the **Newton-Cotes** method approximates the integrand with a polynomial of degree 1 (trapezoid rule), 2 (Simpson) or more and evaluates it at evenly-spaced points in $[a, b]$ (use the Lagrange polynomials $L_i(x) = \prod_{k \neq i} (x - x_k) / (x_i - x_k)$ to find the weights in $\int_a^b f \approx \sum w_i f(x_i)$). The precision and speed of convergence (as the intervals become smaller) can be improved with Richardson extrapolation (Romberg method) or by using the first derivative, if known (extended trapezoid). By choosing both the weights and the subdivision points, **Gaussian quadrature** $\int_a^b f \approx \sum w_i f(x_i)$ can integrate exactly polynomials of degree up to $2n - 1$. This can be generalized to products of polynomials with a given function ϕ (orthogonal polynomials wrt ϕ), and, thus, integrals on an infinite interval (which can also be estimated by just truncating the interval).

The **Laplace transform** can transform some PDEs (e.g., the Black-Scholes one) into ODEs, which can often be solved explicitly. The Laplace transform $\mathcal{L}f(x) = \int_0^\infty e^{-xt} f(t) dt$ is often considered difficult to invert numerically: indeed, its values are weighted averages of the initial function f , and the weights e^{-xt} do

not change much. But if you have an analytic expression for $\mathcal{L}f$, you can recover f , in a numerically stable way, by integrating in the complex plane (Bromwich inversion formula). As usual, convergence can be improved even further with series acceleration methods (Euler summation).

There is also a chapter on copulas and their estimation.

The second part of the book presents 17 case studies.

Random data (bootstrap) can help see the effect of estimation risk in portfolio construction (you do not have the real values of the expected returns and the variance matrix, but just estimations).

Market sensitivity (beta) can be estimated with least median of squares, least trimmed squares, iteratively reweighted least squares (IRLS), shrinkage, bayesian estimates, exponentially weighted least squares, Kalman filter.

One can test technical trading rules (MACD) (the null hypothesis is that the profits of the strategy can be explained by the model, *i.e.*, that they are not significantly different from zero) with a simulated p -value: estimate a reasonable model (GARCH, etc.) for the data, sample from this model N times, look at the corresponding distribution of profits, and compare with the actual profits (from the real, non-simulated data) and conclude.

The risk-neutral density can be estimated from option prices by parametrizing the volatility surface in some way and using

$$q(x) = e^{r\tau} \left. \frac{\partial^2 \text{Call}}{\partial K} \right|_{K=x}$$

but the resulting second derivative may present singularities, fail to be positive, or fail to integrate to 1. Instead, one can look for the density as a mixture of gaussians: the option prices are then weighted sums of Black-Scholes prices, and the problem reduces to a non-linear regression.

American options can be priced with dynamic programming, finding V_t , the value of the option at time t if it has not been exercised yet, by regression $V_t \sim V_{t+1}$ (If $Z_t = \text{Max}(K - S_t, 0)$ is the payoff at time t , V_t is $V_t = \text{Max}(Z_t, e^{-r\Delta} E_t V_{t+\Delta})$ is called the (discrete) **Snell envelope** of Z). Dually, switching from the buyer to the seller, one can compute the optimal hedge (the details are not given):

$$V = \inf_{M \text{ martingale}} S \sup_t (e^{-rt} \text{Payoff}(t) - M_t)$$

The martingale M^* minimizing this quantity is not known, but heuristic choices often give good results and acceptable hedging strategies.

Stochastic volatility models, such as the Heston model

$$dS = \mu S dt + \sigma S dW_1 \\ d\sigma^2 = \alpha(\bar{\sigma}^2 - \sigma^2) dt + \beta \sigma dW_2$$

$$\text{Cor}(dW_1, dW_2) = \rho < 0$$

(prefer the Milstein scheme to simulate it: the Euler scheme does not converge very quickly for square root processes) can account for kurtosis, negative skewness, volatility clustering, but not the price of deep out of the money options: they can be explained by adding jumps to the model. The Fourier transform of the risk-neutral probability density of the log-price at expiry is available, and can be used to calibrate the model (option prices do not have a Fourier transform: multiply them by an exponential damping factor to make them square integrable). The calibration is more numerically stable if one uses implied volatilities instead of prices. Try to minimize several loss functions, e.g., the mean square error, the relative mean square error, or the mean absolute difference.

Here are a few ways of pricing an Asian option (whose payoff depends on the average price)

- Approximate the distribution of $A_T = \frac{1}{T} \int S_t dt$ as log-normal, Edgeworth log-normal, or reciprocal gamma (this is what it is, asymptotically, if prices are log-normal) with the desired first moments (they are easy to compute).
- Use inequalities, e.g.,

$$E[X_+] = E[E[X_+|Z]] \geq E[E[X|Z]_+],$$

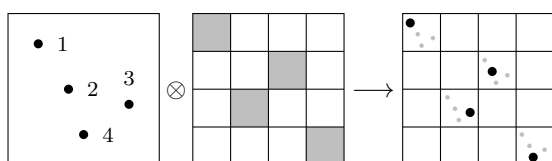
and choose Z so that the difference

$$0 \leq E[X_+] - E[E[X|Z]_+] \leq \frac{1}{2} E\sqrt{\text{Var}(X|Z)}$$

be small.

- Solve the PDE (it is not straightforward to derive, but with a judicious change of numeraire, you get a 1-state-variable PDE).
- The Laplace transform of the option price wrt time can be computed explicitly (either using the PDE, or using the fact that a geometric brownian motion is a time-changed Bessel process).
- You can also use the double Laplace transform, wrt strike and time (or volatility).

A **Latin square** (Latin hypercube) of size n is an $n \times n$ matrix (an n^d array) of zeroes and ones, with exactly one 1 in each row and column (in each hyperplane). To select a Latin square at random, just take two permutations $\sigma_1, \sigma_2 \in \mathfrak{S}_n$, set $a_{\sigma_1(i), \sigma_2(i)} = 1$ and leave the other elements to zero. To sample n points in $[0, 1]^2$, using a Latin square to better spread them out, take n points at random in $[0, 1]^2$, take a Latin square at random, translate and rescale the random points to put them in each cell of the Latin square.



In dimension 1, a low discrepancy sequence (van der Corput) can be constructed as follows:

- Write increasing integers (0, 1, 2, 3, etc.) in base b (0, 1, 10, 11, 100, etc.);
- Write the digits in reverse order and add a decimal point in front of them (0, .1, .01, .11, .001, etc; *i.e.*, $0, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}$, etc.).

This is not unlike the Farey sequence ($0, 1, \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, \frac{1}{7}$, etc.), in number theory Sobol sequences generalize this idea in higher dimensions. To estimate the error of a quasi Monte Carlo computation using statistical methods, you may want to scramble the sequence first (to make it random).

The discrete maximum of an (arithmetic) Brownian motion (used by look-back options) can be computed using the z -transform.

Agent-based models can be used to describe **electricity prices** – the stylized facts are different from those of stock prices: seasonality, spikes, strong mean reversion, time-dependent volatility. The times series models used to price options on energy prices (these are real options: the optionality is the ability to switch the plant or generator on) have to account for those stylized facts. Swing contracts (a kind of American option, where you can buy a commodity at a given price, in an arbitrary quantity, every day in the year) can be priced with dynamic programming.

Mortgage prepayment is important to value MBS (mortgage-backed securities): borrowers with an improving credit rating are more likely to pre-pay their debt earlier to switch to lower rates – leaving only bad borrowers. Econometric forecast of the prepayment, using historical data, does not perform well, but the behaviour of the borrower, choosing to switch to new rates or not, can be modelled as an American option, and the optimal prepayment strategy can be computed via dynamic programming.

Structural models of default, used to price CDS (credit default swaps), compare the enterprise value (a geometric Brownian motion or some other stochastic process) with the debt of the company

$$\tau = \inf\{t : \text{Enterprise Value}_t \leq \text{Debt}\}.$$

Reduced form models of default directly model the default time, with an intensity process. For basket default swaps, one usually mixes those default times with a 1-factor Gaussian copula (the factor can be interpreted (chosen?) as the business cycle, some stock index, interest rates, etc.).

Monte Carlo simulations, in high dimensions, can be sped up and simplified as follows:

- Perform a principal component analysis (PCA) on the data to simulate, to reduce the dimension;
- Replace the continuous distribution (of the factors retained from the PCA) with a small number (10 to 2, depending on the importance of the factor) of points or “scenarios”;
- Sample from the resulting (discrete) distribution,

To estimate the parameters of a diffusion (or a jump-diffusion) via maximum likelihood, you need the transition probabilities. They can be estimated via Monte Carlo simulation: just split the intervals between two observations into smaller intervals, and simulate the diffusion using the Euler scheme. (Simulated maximum likelihood, SML)

The drift and diffusion coefficients of a diffusion process are its instantaneous first moments – higher moments are zero. Jump diffusion processes produce non-zero higher instantaneous moments.

$$dX = \mu(X)dt + \sigma(X)dW + dJ$$

$$\mu(x) = \lim_{h \rightarrow 0} \frac{1}{h} E[X_{t+h} - X_t | X_t = x]$$

$$\sigma^2(x) + \lambda(x)E[Y^2] = \lim_{h \rightarrow 0} \frac{1}{h} E[(X_{t+h} - X_t)^2 | X_t = x]$$

$$\lambda(x)E[Y^j] = \lim_{h \rightarrow 0} \frac{1}{h} E[(X_{t+h} - X_t)^j | X_t = x]$$

Non-parametric (kernel) estimators of those moments give non-parametric estimates of drift μ , diffusion σ , jump intensity λ and jump distribution Y (you have to make some more assumptions on the distribution of jumps, e.g. Gaussian with zero mean).

GARCH models (and their generalizations) can be estimated from the volatility surface, by considering it as a moment (in the generalized method of moments, GMM).

***The accountant's guide to the universe
Heaven and hell by the numbers
C. Hovey (2010)***

Could accounting be made more interesting by applying it to non-conventional enterprises, such as crime, or deciding whether one should go to heaven or hell? The book presents several economic and accounting notions (bounded rationality, free markets, creative destruction, present value, double-entry book-keeping, balance sheet, income statement) for those two examples. The result is pleasantly funny, but you will not learn anything: the notions of credit and debit are not explained (I vaguely remember from my accounting courses that “debit” means the opposite of what it means for my bank account – at least, half the time), some of the numeric examples are wrong (the publisher dropped some of the credit or debit columns) or have to be accepted on faith (present value computations), and the reluctance to use negative numbers complicates things (you never know in which direction the money is going).

Computer scientists will prefer M. Kleppmann's graph-theoretic explanations.

***Quantifying and modeling
long-range cross-correlations
in multiple time series
with applications to world stock indices
D. Wang et al. (2011)***

Random matrix theory (RMT, comparing the distribution of the eigenvalues of a sample correlation matrix with the distribution coming from the sample correlation matrix of iid gaussian variables) can also be applied to cross-correlation. In a financial context, you should look at both the correlation of returns and absolute returns – the latter decay more slowly. Using a 1-factor (sic) model, the authors find 10 uncorrelated indices that may form a low-risk portfolio. (Other methods could be used, e.g., detrended cross-correlation analysis, DCCA).

***Predicted and verified
deviations from Zipf's law
in ecology of competing products
R. Hisano et al. (2011)***

Zipf's law (stable distribution with power law tails) appears in the following situation:

- A finite population of “entities” I_t changes with time, with entities entering or leaving the universe according to a Poisson process;
- Each entity $i \in I_t$ is described by a geometric Brownian motion $S_i(t)$, starting with some random value at time t_i (when i enters I);
- Then, for $t \gg 1$ fixed (and $|I_t| \gg 1$), the distribution of the $S_i(t)$, $i \in I_t$, has power law tails.

***Utility theory front to back
inferring utility from agents' choices
A.M.G. Cox et al.***

The optimal investment problem often assumes that the utility function is given, but determining which utility function faithfully describes the preferences of a given investor is tricky, and usually done by looking at her choices when faced with 1-period investment problems. This article tries to infer the utility function from the actual (observed) consumption choices in a continuous-time framework (deterministic or stochastic (Black-Scholes)). Not all consumption choices come from a utility function (they have to satisfy some PDE), and the utility function (or, even, the sign of the relative risk aversion) is not entirely determined by the consumption choices.

***Topological isomorphism
of human brain and financial market networks
P.E. V ertes et al. (2011)***

The growing graph, minimum spanning tree and the various graph-theoretical measures (modularity, etc.) of (the variance matrix of) fMRI and financial data are very similar.

***The common component
of idiosyncratic volatility
J. Duarte et al. (2011)***

One more factor should be added to the Fama-French model

returns \sim market + size + book-to-market,

corresponding to the “idiosyncratic volatility” (the first principal component of the variance matrix of the residual return – it is always a good idea to add statistical factors to your risk model, to catch anything you may have overlooked, or anything that may have changed); it can be interpreted as the position in the business cycle.

*An empirical analysis
of dynamic multiscale hedging
using wavelet decomposition*
T. Conlon and J. Cotter (2011)

You can assess the influence of the investment horizon on the minimum variance portfolio (or mean-variance, or mean-Gini – **Gini’s mean difference** is $E[\frac{1}{2}|X - Y|] = 2 \text{Cov}(X, F_X(X))$, where $X \perp\!\!\!\perp Y$ are distributed as the portfolio returns) by looking at the wavelet transforms of the returns time series and expressing the variance matrix, the portfolio weights, the higher moments of the portfolio, etc., as functions of the wavelet coefficients (maximum overlap discrete wavelet transform). Only the “optimal hedge ratio” case, *i.e.*, the 2-asset case, is treated (one asset is the portfolio you hold and want to hedge, its weight is imposed to be 1, the other is the hedge portfolio, already computed, whose weight has to be determined).

Anchor modeling
L. Rönnbäck et al. (2010)

Anchor modeling is a rebranding of entity-relation (ER) modeling, that describes a database schema in terms of actors (entities), attributes (historicized or not), knots (dimension tables, *i.e.*, attributes with a small or fixed number of values), ties (relations) and, more importantly, advocates that changes to the database be implemented as extensions rather than modifications – this ensures backward compatibility.

*Financial models with Levy processes
and volatility clustering*
S.T. Rachev et al. (2011)

The main claim of the book is that non-gaussian distributions should and can be used in practice, to model stock prices, to price options, to optimize portfolios – even if those distributions do not have an easy-to-use probability distribution function, their characteristic function is simple, and sufficient for all practical purposes – at worst, you have to compute one or two integrals, via the FFT (Fast Fourier Transform), quadratures, or Monte Carlo simulations).

(Beware: the book has apparently not been proof-read and contains innumerable incorrect formulas; many explanations are also seriously lacking – the original articles are often clearer.)

The gamma distribution $\Gamma(c, \lambda)$ is the law of $Y_1 + \dots + Y_c$ where $Y_i \sim \text{Exp}(\lambda)$ iid. The **variance gamma** distribution $\text{VG}(c, \lambda_+, \lambda_-)$ is the law of $G_+ - G_-$ where $G_{\pm} \sim \Gamma(c, \lambda_{\pm})$. The **inverse Gaussian** distribution

is the law of the first time at which a random walk reaches a given value $\tau = \text{Min}\{t : X_t = m\}$.

Stable (or α -stable) distributions can be defined by (and used through) their characteristic function. One can define many variants of the stable distributions, *e.g.*, by cutting the tails and replacing them with Gaussian tails (truncated stable distributions) or by tweaking the characteristic function (tempered stable distributions) or (equivalently) by multiplying the Lévy measure of the α -stable distribution (α -stable distributions are infinitely divisible, and can therefore be expressed using the Lévy-Khinchine formula) by some tempering function (*e.g.*, e^{-t}).

(Tempered) α -stable distributions can be fitted either by fitting their characteristic function, or via maximum likelihood, using a numerical approximation of their density function.

A **compound Poisson process** is a process of the form $X_t = \sum_{k=1}^{N_t} Y_k$, where the Y_k are iid (arbitrary distribution) and $N_t \sim \text{Pois}(t\lambda)$ is a Poisson process. A **pure jump process** is a limit of compound Poisson processes: let $S \subset \mathbf{R}$ be a countable set (jump sizes), $\lambda : S \rightarrow \mathbf{R}_+$ some function, $N_t^{\lambda(s)}$ Poisson processes of intensity $\lambda(s)$, and $Y_t = \gamma t + \sum_{s \in S} s N_t^{\lambda(s)}$. In the expression of the characteristic function of Y_t , one can replace the measure

$$\nu(A) = \sum_{s \in S \cap A} \lambda(s)$$

(expected number of jumps of size in A per unit of time) with a more general measure – it is the Lévy measure of the pure jump process.

A non-Gaussian infinitely divisible random variable X_1 (*e.g.*, gamma, α -stable, inverse gaussian, variance gamma) defines a pure jump process.

Non-decreasing pure jump processes (gamma, inverse gaussian, Poisson) can be interpreted as a “cumulated trading activity” and used to define time-changed Brownian motions: the inverse gaussian gives the **normal inverse gaussian** (NIG); the gamma distribution gives the **variance gamma** process.

A **Lévy process** is a process with independent stationary increments, càdlàg and stochastically continuous

$$\forall t \quad \forall \varepsilon > 0 \quad \lim_{s \rightarrow t} P[|X_s - X_t|] = 0.$$

It is the sum of a Brownian motion and a pure jump process. An exponential Lévy process is the exponential of a Lévy process.

It is possible to simulate a random variable from its characteristic function (if the tails are not too fat), using

$$f(x) \leq \frac{1}{2\pi} \int_{-\infty}^{\infty} |\phi| \\ f(x) \leq \frac{1}{2\pi x^2} \int_{-\infty}^{\infty} |\phi''|,$$

by rejection – it is easy to sample from

$$g(x) \propto \text{Min} \left(\int_{-\infty}^{\infty} |\phi|, \frac{1}{x^2} \int_{-\infty}^{\infty} |\phi''| \right).$$

Lévy processes can be simulated, by writing them as

$$X_t = Y_t^\varepsilon + Z_t^\varepsilon,$$

where Z_t^ε is the compound Poisson process containing the jumps of size larger than ε , and Y_t^ε is either neglected or replaced by an easy-to-simulate process (e.g., a Brownian motion). There are ad hoc algorithms to simulate from the tempered α -stable distributions and a few special cases (NIG, VG, etc.).

The Girsanov theorem links the (known) real-world probability \mathbf{P} to the risk-neutral one \mathbf{Q} (needed to compute option prices).

For elliptical distributions,

$$f(x) \propto g((x - \mu)' \Sigma^{-1} (x - \mu)),$$

the correlation completely describes the dependence. Elliptical distributions can account for the fat tails and tail-dependence in asset returns but, because of radial symmetry, not for the larger dependence in the lower tail. Elliptic distributions can be written as

$$X = \mu + RAS$$

where $S \sim U(\mathbf{S}^{d-1})$, R is a random variable in \mathbf{R}^d and A is a $d \times d$ matrix, with $R \perp\!\!\!\perp AS$. Instead, one can let the distribution of R depend on the direction AS : let $s : \mathbf{R}^d \rightarrow [0, 1]$ be a function, measuring the tail dependence in a given direction, $(R_t)_{t \in [0, 1]}$ a family of positive random variables, and consider the **multi-tail generalized elliptic distribution**

$$X = \mu + R_{s(AS)} AS.$$

The tail function s could come, for instance, from the first principal component v_1 of $\Sigma = AA'$,

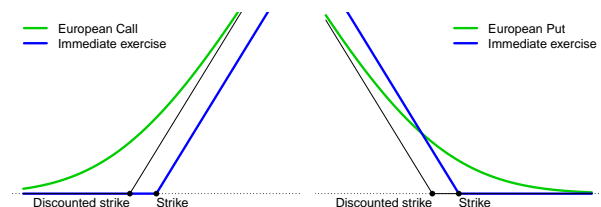
$$s(x) = \alpha_+ \mathbf{1}_{\langle x, v_1 \rangle > 0}(x) + \alpha_- \mathbf{1}_{\langle x, v_1 \rangle < 0}(x).$$

Portfolios of non-gaussian assets can be optimized with a scenario-based approach (fit the data, simulate returns, optimize of those scenarios), which also allows for more flexibility the measure of risk.

Option prices can be computed for tempered stable processes, but some integrals remain – there are convergence problems with (non-tempered) stable processes: that is why **tempered** stable distributions were introduced.

Options can also be priced with GARCH models (e.g., with truncated stable innovations), in discrete time, but more technical complications emerge (non-unique risk-free measure, etc.). GARCH parameters, when used for option pricing, are usually estimated from option prices, completely ignoring the underlier – models with non-gaussian residuals may be able to reconcile the underlier and the options.

The following pictures explain why early exercise of an American call option is never optimal (it is always more profitable to sell the American option as a European one than to exercise it) and why early exercise of an American put option is sometimes optimal (if it were not, the price would be the same as that of a European put, but that is not possible because, for a deep-in-the-money option, it is more profitable to exercise it immediately rather than sell it at the same price as a European option – however, this does not tell you *when* to exercise the option).



Mixed effects models and extensions in ecology with R

A.F. Zuur et al. (Springer, 2009)

This book explains, with no mathematical overhead, what to do when the assumptions of the linear regression model (linearity, gaussian noise, homoscedasticity, independence – they can be checked by plotting the residuals versus the fitted values, the predictors, and with a Gaussian quantile-quantile plot) are not satisfied.

Additive models use splines or local regression to account for non-linearity.

```
gam: : gam(y~loess(x, span=.5))
mgcv: : gam(y~s(x, fx=FALSE, k=-1, bs="cr"))
mgcv: : gam(y~s(x)+s(x, by=g==1)+factor(g))
```

Generalized least squares (GLS) can take care of heteroscedasticity (you can also transform the data); there are many specifications of the variance (group-dependent, linear in one of the predictors, etc.); you can check that the heteroscedasticity has been removed by plotting the normalized residuals.

```
nlme: : lme(y~x*g, weights=varIdent(form=~1|g))
```

When the observations are grouped, you may be tempted by a 2-stage approach: first compute a regression in each group, then study the distribution of the corresponding regression parameters (either fit their distribution, or regress them against group-specific variables). Mixed models are a more efficient 1-step procedure.

```
# Random effects
nlme: : lme(y~1, random=~1|g))
# Random intercept
nlme: : lme(y~x, random=~1|g))
# Random intercept and slope
nlme: : lme(y~x, random=~1+x|g))
```

It turns out that mixed models can be described in two equivalent ways: either a procedural way, as above (à la Bugs), or by just specifying the shape of the variance

matrix, as in a GLS model (for instance, the random effects model corresponds to a block-diagonal correlation matrix, with ρ in all the off-diagonal entries of the diagonal blocks).

```
gls(y~x,
    correlation=corCompSymm(form=~1|g),
    method="REML")
```

This also works with splines

```
mgcv::gamm(y~x1+s(x2),random=list(g=~1)),
temporal correlation (beware of missing values: if you
remove holes in a time series, i.e., skip values, the auto-
correlation plots will be incorrect)
```

```
gls(y~x,correlation=corAR1(form=~t))
```

```
gls(y~x,
    correlation=corARMA(
        c(.3,-.3), # starting values
        p=2,q=0,form=~t))
```

```
mgcv::gamm(y~x,
    correlation=corAR1(form=~t|g),
    weights=varIdent(form=~1|g)),
temporal correlation with irregularly-spaced observa-
tions
```

```
gls(y~x,correlation=corGaus(form=~t|g))
```

or spacial correlation

```
gls(z~u,correlation=
    corGaus(form=~x+y,nugget=TRUE)).
```

Spacial correlation manifests itself by clusters of resid-
uals. The variogram,

$$\gamma(\mathbf{h}) = \frac{1}{2} \text{Var}[Z(\mathbf{x} + \mathbf{h}) - Z(\mathbf{x})]$$

is useful for weakly stationary and isotropic data:
`gstat::bubble`, `gstat::variogram`, `nlme::Vario-`
`gram`, `geoR::variog`.

The restricted maximum likelihood (**REML**) gives un-
biased variance estimators. Even for linear regression,
 $Y = X\beta + \varepsilon$, the maximum likelihood estimator of the
variance is biased – the more parameters, the larger
the bias. However, if A is a matrix such that $AX = 0$,
then $AY = A\varepsilon$, *i.e.*, $AY \sim N(0, \sigma^2 AA')$, gives an unbi-
ased estimator of the variance σ^2 . The same procedure
can be applied to a mixed model, $Y = X\beta + Zb + \varepsilon$,
 $b_i \sim N(0, D)$, $\varepsilon_i \sim N(0, \Sigma_i)$, which can be written as
 $Y_i \sim N(X_i\beta, V_i)$, where $V_i = Z_i D Z_i' + \Sigma_i$: if $AX = 0$,
then $Y_i \sim N(0, AV_i A')$ can be used to estimate the
variance.

The model selection procedure goes as follows: in-
clude as many fixed effects and interaction terms as
possible; determine the random components (REML);
determine the fixed components (ML); fit the model
(REML). Use the following functions: `drop1`, `step`,
`anova`, `lmtest::lrtest`.

Count data are not gaussian and can be modeled with
a Poisson ($\text{Var } X = EX$), negative binomial (number
of successes until k failures in a series of independent
Bernoulli trials, overdispersed: $\text{Var } x \geq EX$), or bino-
mial (count data in $\llbracket 0, n \rrbracket$, binary data) distribution.
These belong to the natural exponential family,

$$f(x, \theta) = h(x) \exp(\theta \cdot x - A(\theta)),$$

a special case of the exponential family

$$f(x, \theta) = h(x) \exp(\eta(\theta) \cdot T(x) - A(\theta)).$$

To use those distributions in a generalized linear model
(GLM), start with the description of the maximum-
likelihood estimation of the linear model

$$\eta_i = \sum_j \beta_j x_{ij}$$

$$y_i \sim N(\eta_i, \sigma)$$

and replace the gaussian distribution $y_i \sim N(\eta_i, \sigma)$
with one of the following.

$$y_i \sim \text{Pois}(\exp \eta_i)$$

$$y_i \sim \text{NB}(\exp \eta_i, k)$$

$$y_i \sim \text{Binom}(1, \text{logit}^{-1} \eta_i)$$

$$y_i \sim \text{Binom}(n_i, \text{logit}^{-1} \eta_i)$$

If there is some over-dispersion, but not too much,
you can tweak the likelihood to account for it (quasi-
maximum-likelihood).

```
glm(y~x, family=poisson)
```

```
glm(y~x, family=quasipoisson)
```

```
MASS::glm.nb(y~x,link="log")
```

```
mgcv::gam(y~s(x),family=negative.binomial(1))
```

```
mgcv::gam(y~s(x),family=binomial)
```

```
glm(y~x,family=binomial) # logit
```

```
glm(y~x,family=quasibinomial)
```

```
VGAM::vglm(y~x,family=negbinomial)
```

For binary data, prefer the clog-log link to the logit
or probit if the numbers of zeroes and ones are very
different.

Those models can be extended to zero-truncated (re-
move the zeroes), zero-inflated (mixture of zero and a
Poisson or negative binomial distribution), zero-altered
(aka hurdle or 2-part: mixture of zero and a zero-
truncated distribution) distributions.

```
VGAM::vglm(y~x,family=posnegbinomial)
```

```
pscl::zeroinfl(y~x|g,
```

```
    dist="poisson",link="logit")
```

```
pscl::hurdle(y~x|g,
```

```
    dist="negbin",link="logit")
```

The opposite of the log-likelihood, to be minimized, is
a sum; by adding some constant (the log-likelihood of
a saturated model), we can ensure that the terms are
positive, the deviances – we are minimizing a sum of
“squares”.

The generalized linear model (GLM) minimizes the
sum of deviance residuals; if, instead, we minimize the
sum of Pearson residuals, we can modify it to account
for dependency between observations, as with mixed
models: this method is called **generalized estimat-**
ing equations (GEE) (the equations in question are
the first order conditions of the minimization problem:
the derivatives of the modified sum of squared residu-
als, wrt to the parameters to estimate, should be zero).

```
geepack::geeglm(y~x,
```

```
    family=poisson,id=g,corstr="ar1")
```

It is possible, but more complicated, to merge mixed models and generalized linear models (in the log-likelihood, you have to integrate out the random effects, e.g., with quadratures or Monte Carlo simulations or simpler/faster but more biased methods).

```
MASS::glmPQL(y~x,random=~1|g,
  family=binomial)
lme4::lmer(y~x+(1|g),family=binomial)
glmmML::glmmML(y~x,cluster=g,family=binomial)
mgcv::gamm(y~s(x),random=list(g=-1),
  family=poisson)
library(help=BRugs)
```

All about high-frequency trading M. Durbin (McGraw Hill, 2010)

High-frequency traders are the descendants of yesterday's market-makers, trying to profit from the bid-ask spread, but the notion now encompasses all strategies relying on execution speed, many of which are based on the market micro-structure (the order book).

High-frequency trading (HFT) typically combines several asset classes (stocks, ETFs, futures, options) and many exchanges and alternative trading systems (ATS, the best known being the ECN, and the various dark pools – contrary to exchanges, they do not display quotes). Passive traders send **limit orders** (executed at a given price, if and when there is a counterparty) while active traders submit **market orders** (executed immediately against the available limit orders). A **quote** is the datum of the current best bid (highest limit buy order), best ask (lowest limit sell order) and the corresponding volumes. The depth of the order book (other limit orders, with their volumes) is also of interest.

There are several types of traders: the **investor** (aka buy-side, *i.e.*, they buy, and only sell as a consequence, sometimes called liquidity-taker); the **market-maker** (aka sell-side, specialist, liquidity-provider), who uses mainly limit orders, and earns the spread between the bid and the ask (he buys at the bid and hopes to sell at the offer shortly after) – there is some risk because the prices do change: market makers will reduce the spread to reduce this risk; the **arbitrageur** uses relations known to hold between the prices of different assets, for instance, the price of a stock should be the same on all exchanges, the put and call option prices should be related through the put-call parity, etc.; the predictor is similar to the arbitrageur, but relies on relations that hold on average, over time (statistical arbitrage), e.g., pairs trading. The high-frequency trader is just a market-maker; he is on the sell-side; the **algorithmic trader** is his counterpart on the buy-side, and tries to minimize market impact. Some markets impose a “minimum spread”: there is a rebate for limit orders and a fee for market orders; this is a kind of “minimum revenue” for market-makers.

Here are some investor strategies:

- Join the makers: send a limit order at the current

bid or ask, and wait for market orders to come, especially if there is a strong imbalance between the bid and ask volumes – over the long time, you will get the VWAP.

- Poke for bargains: send a (limit) order inside the spread, and see if there is any reaction – market makers may be willing to trade at a lower spread than currently displayed – some exchanges allow hidden limit orders, called reserve orders – and repeat with a price closer to the bid or ask until someone reacts.
- Iceberg order: limit order whose large size is hidden (you do not have to do it yourself: the exchange can do it for you, and only display x shares at a time).
- Time slicing: similar to the iceberg order, but wait some random time before displaying the next slice.
- Out-smart the market-makers: coax the market-makers into following one of their well-known strategies (e.g., by mimicking a sell iceberg when you actually want to buy).

Here are a few market-maker strategies:

- Wait for the other side: if you bought at the bid, send a limit sell order at the ask and wait, hoping that the price does not change.
- Lean your market: instead of sending a limit sell order at the ask, you can lower the ask, reducing the spread and your potential benefit, but increasing your chances of seeing the order filled.
- Scratch the rebate: in markets with a rebate for limit orders, you can reduce the ask down to the bid and still earn the rebate.
- Hide your best prices: keep the spread wide, wait for limit orders, and fill them with market orders (you do not earn the rebate). Market-makers have therefore two spreads: a wide passive spread, and a narrower active spread.
- Take out slow movers: market-makers' spreads moves as the prices move; a market-maker's active spread may cross the passive spread of a slower market-maker.
- Penny jump: if there is a big order in the order book, put a smaller one just before: the big order will limit your losses (if you buy the shares and the price moves in the wrong direction, you can sell them back).
- Push the elephant: when a large buy limit order comes, you may suspect a “join-the-makers” strategy, *i.e.*, that the investor is willing to pay slightly more; you can try to increase the price, by raising the ask (buy a small quantity at a higher price); the elephant may raise his bid a few times, after which you can profit from him. This is a kind of front-running, but using only public information (probably legal); it is also a form of price manipulation (which may or may not be legal).
- Tow the iceberg: same strategy, with an iceberg (hidden elephant).
- Delta-hedging: Market-maker strategies for stocks involve a round-trip: you buy at some price and sell the same asset later at another price (or the opposite). For options, because of the large number of contracts, this may not be possible: instead of com-

pleting the round-trip, you may simply delta-hedge.

- Jump the delta: Look for large option trades, estimate the amount of stock needed to hedge the position, expect a trade in the underlying, and try to get in front of it (to avoid this front-running, market-makers may prefer to hedge with an index instead of the underlier).

Here are some arbitrage and statistical arbitrage strategies:

- ETF versus basket, futures versus basket, futures versus ETF, futures versus futures (on the same underlying, with different lot sizes);
- Futures versus options (put-call parity);
- Volatility arbitrage: option prices or, equivalently, implied volatility, should be a smooth function of strike and time-to-expiry: some contracts may be over- or under-priced;
- Spread arbitrage: some composite options, such as spreads, strangles, straddles, butterflies, are quoted, and their price may differ from that suggested by puts and calls;
- Pairs trading: estimate the fair price or “micro-price” of a stock, as mid-price + α , and look for bids or asks on the wrong side of the micro-price.
- Futures lag (not explained clearly): try to estimate how much each stock of an index lags behind the futures;
- Event-driven: when a dividend is announced, the price will jump or drop almost immediately, but you may have time to buy or sell just before (or, at least, before it moves all the way); dividends also impact option prices, directly (they appear in the formula), and indirectly (through the price changes);
- Trend-following, e.g., you can compare 5-minute and 5-hour moving averages and buy or sell when they cross;
- Mean-reversion: the market tends to over-react, e.g. to imbalances in the order book.

The IT infrastructure is geared towards performance (C++, distributed programming, load balancing, TCP/IP off-load engine (TOE), GPU, parallel processing – no mention of FPGA), relies on good programming practices (design for change), an even-driven architecture (“pub-sub”, *i.e.*, “publish-subscribe”, either in hardware (TMX Message Switch) or in software (RendezVous, by Tibco)), and direct market access through collocation (put your machines in the same building as the exchange) and the use of native APIs (each exchange has its own API, faster than the FIX standard).

It has the following components:

- Thinkers contain the strategies, their parameters, the universe (set of assets), the compliance rules.
- Listeners receive data from the exchange, filter it for erroneous or unnecessary data (large changes, small quantities, changes that immediately revert), keep track of the state of the market (open/closed, high/low volume, etc.).
- Pricers compute prices, alphas, micro-prices (price +

α + correction for the inventory we have), bids, asks; for options, the implied volatility (volatility curve or surface, corrected for inventory) is first computed, and then used to compute the option prices (if the model is complicated, they may be precomputed), bids and asks.

- Traders send trades to the exchange; they usually ask the exchange to perform some safety controls (automatically remove the quotes if there is too much activity), and check the heartbeat of the system (remove all the quotes if the exchange does not receive the next heartbeat). The passive trading (limit orders) part is called the “quoting engine” and the active trading (market or IOC (immediate-or-cancel) orders) part is the “electronic eye”. There is also a component to locate assets we may want to short, and another one to perform (delta-, and also rho-) hedging.
- Managers keep track of the positions (inventory), look out for bad trades, measure risk (exposure, *i.e.*, size of the positions, Greeks, what-if analyses, etc.), compute the P&L (the theoretical (“front office”) one, using the micro-prices, used for trading, and the “mark-to-market” (“back-office”) one, using actual market prices), ask the traders to stop trading or reduce their position if some risk limits are exceeded, completely stops the system in case of a problem (the connection to the exchange was lost, one of the components crashed, etc.).

High-frequency trading (HFT) leads to lower bid-offer spreads, better price consistency (faster information transfer between various exchanges), and is not subject to human panics (I do not entirely agree on this point: the failure mode is just different, but can be as dramatic); however, many strategies are akin to price manipulation, increase volatility (I partly disagree: HFT stabilizes volatility, but maintains it above a certain level), bad trades can lead to contagion and cascade effects, most of the trading now seems to be unrelated to the underlying companies.

Securities lending, shorting and pricing **D. Duffie et al. (2002)**

Short selling has an impact on prices: the price is the value of the stock plus that of the option to lend it to short sellers. The value of the option depends on supply, demand, and the bargaining power balance between lenders and borrowers.

Portfolio consumption choice with stochastic investment opportunities and habit formation in preferences **C. Munk (2007)**

To ensure that future consumption does not fall below the habit level, increase the weight of bonds reduce that of stocks.

Dynamic asset allocation with stochastic income and interest rates

C. Munk and C. Sørensen (2009)

Considering labour income in optimal long-term portfolio choice seems to suggest that labour income is equivalent to a bond, so that you can reduce or completely remove bonds from your portfolio. This is no longer true if income or interest rates are stochastic and dependent, e.g., if your income (employment status) fluctuates with the business cycle.

***The variance gamma process
and option pricing*
D.B. Madan et al. (1998)**

A gamma process is an infinitely divisible process X_t with $X_1 \sim \text{Gamma}$.

A variance gamma process is a brownian motion with drift evaluated at random times given by a gamma process (a subordinated process). Since it has finite variation, it can be decomposed as the difference of two non-decreasing (gamma) processes.

***Modelling the spacial dynamics
of culture spreading
in the presence of culture strongholds*
L. Lizana et al. (2011)**

Information spreads on a network (or a lattice), randomly, from a central source, with new information replacing old information, creating a concentric wave pattern. The same phenomenon also describes word distribution, newly-coined words replacing older ones – for instance, Japanese swear words (ばか, あほ, etc.) seem to emanate from the old capital, Kyōto.

***Temporal patterns of happiness and
information in a global social network:
hedonometrics and twitter*
P.S. Dodds et al. (2011)**

For each tweet, the authors compute:

- A happiness level, between 0 and 10, defined as a weighted average of the happiness level of the words it contains – the words were assigned a happiness level manually, using volunteers (or underpaid workers) at Amazon’s Turk. (The article suggests that the data is available online.) Stopwords should be removed: they muddy the signal.
- The word diversity, measured by entropy $\sum p \log p$ or some of its generalizations $\sum p^\alpha$ (e.g., $\alpha = 2$ is related to the Gini coefficient); it can be converted into an “effective number of words” (the number of words yielding the same entropy if they all have the same frequency). Word diversity and happiness seem unrelated.

One can look at seasonal changes (there are weekly and daily patterns), the effects of some events (Christmas, financial crises, etc.), or compare texts that contain a given word or expression with the background happiness and word diversity. The differences are displayed in a *word shift graph*, that shows the words with the largest contributions to the change in happiness, and whether it is an increase or a decrease in the frequency of a positive or negative word.

The lack of context may pose problem: “children” is a positive word, but when it appears together with “victim”, it (incorrectly) reduces the negativity. Since words are not stemmed, there can be an increase in “dead” and a decrease in “die” (this may be meaningful: it is a change in tense).

Classical optimization studies problems of the form

$$\begin{aligned} &\text{Minimize } \phi(x) \\ &\text{Where } x \in K \end{aligned}$$

If ϕ is \mathcal{C}^0 and K compact, the problem has a solution. The continuity condition can be relaxed to **lower semi-continuity**, *i.e.*,

$$\liminf_{x \rightarrow x_0} \phi(x) \geq \phi(x_0)$$

instead of

$$\lim_{x \rightarrow x_0} \phi(x) = \phi(x_0),$$

i.e., there can be some noise in the function, but it must be “upwards”, so as not to affect the minimization problem.

If ϕ is differentiable and $K = \mathbf{R}^n$, a necessary condition for x to be a solution is that the gradient in x be zero: $D\phi = 0$. If the problem is constrained, *i.e.*, $K \subsetneq \mathbf{R}^n$, the condition becomes

$$\forall y \in T_x K \quad D\phi(x) \cdot y \geq 0$$

where $T_x K$ is the tangent cone of K at x (in particular, it is \mathbf{R}^n if x is interior). If, in addition,

$$\forall h \in T_x K \setminus \{0\} \quad D^2\phi(x)(h, h) > 0$$

then the point is a (local) minimum.

The **Kuhn and Tucker condition** (aka **Lagrange multipliers**) translate these conditions into useable formulas (by explicitly describing the tangent cone) when the set of admissible solutions is of the form:

$$K = \{x \in \mathbf{R}^n : \forall i \in \llbracket 1, p \rrbracket \quad b_i(x) \leq 0\}.$$

The **Lagrangian** is

$$L(x, \lambda) = \phi(x) + \lambda \cdot b(x).$$

If x is a solution, then there exists $\lambda \in \mathbf{R}^n$ such that

$$\begin{aligned} DL(x, \lambda) &= 0 \\ \lambda \cdot b(x) &= 0. \end{aligned}$$

(You may need to remove a few pathologies, such as linear dependencies between the $Db_i(x)$.)

Calculus of variations is still interested in minimization problems, but this time, we are looking for a function that minimizes some integral:

$$\begin{aligned} &\text{Find } x : [t_0, t_1] \longrightarrow \mathbf{R}^n \mathcal{C}^1 \\ &\text{to minimize } \int_{t_0}^{t_1} F(t, x(t), \dot{x}(t)) dt \\ &\text{such that } x(t_0) = x_0 \text{ and } x(t_1) = x_1. \end{aligned}$$

(The final condition can be replaced by a set of equalities or inequalities on the coordinates of $x(t_1)$.) Penalized regression can be formulated in this way. The

local Euler equation is a necessary condition, in the case of equality constraints ($x(t_1) = x_1$):

$$\frac{d}{dt} \frac{\partial F}{\partial \dot{x}} = \frac{\partial F}{\partial x}.$$

(To prove it, assume that x is a solution, take a \mathcal{C}^1 function h with $h(t_0) = h(t_1) = 0$, consider the cost of $x + \varepsilon h$, integrate by parts.) This condition remains valid if you are looking for a piecewise \mathcal{C}^1 function instead, provided you write it as an integral

$$\frac{\partial F}{\partial \dot{x}} = \int \frac{\partial F}{\partial x} dt + \text{constant}.$$

The constraints $x(t_1)_i = x_{1i}$ or $x(t_1)_j \geq x_{1j}$ or $x(t_1)_k \leq x_{1k}$ lead to the **transversality conditions**: $\frac{\partial F}{\partial \dot{x}_i} \geq 0$ or $\frac{\partial F}{\partial \dot{x}_i} \leq 0$ if the condition is binding (for a minimization problem, the inequality is in the same direction as the constraint); $\frac{\partial F}{\partial \dot{x}_i} = 0$ if the condition is not binding or if there is no constraint (there is no transversality condition for equality constraints).

If the $F(t, \cdot, \cdot)$ are convex for all t , then those conditions are sufficient.

A **Lagrange problem** is a generalization of calculus of variation:

$$\begin{aligned} &\text{Find } u \\ &\text{to minimize } \int_{t_0}^{t_1} F(t, x(t), u(t)) dt \\ &\text{where } \dot{x}(t) = f(t, x(t), u(t)) \\ &\text{and } x(t_0) = x_0 \end{aligned}$$

Since u controls x via the ODE, this is sometimes called an **optimal control** problem. There are equivalent formulations where the cost function only depends on the final value of x (Mayer formulation). One usually imposes some conditions on f to avoid explosions, *e.g.*, Lipschitz and linear growth. The **Hamiltonian** of the problem is

$$F(t, x, u, p) = F(t, x, u) + p \cdot f(t, x, u).$$

If u is a solution of the Lagrange problem, then there exists a function p , called the adjoint state, such that

$$\begin{aligned} \dot{p}(t) &= -\frac{\partial H}{\partial x}(t, x(t), u(t), p(t)) \\ p(t_1) &= 0 \text{ (transversality condition)} \\ H(t, x(t), u(t), p(t)) &= \min_v H(t, x(t), v, p(t)). \end{aligned}$$

These equations (if you want to solve them, also add $\dot{x} = \partial H / \partial p$, which comes from the definition of H) are called the **Pontryagin principle** (or the hamiltonian system, if you remove the one with the minimum). By adding a few more conditions, you can have a set of sufficient conditions.

One can use dynamic programming to solve the same (Lagrange) problem. Let $J(t, \xi, u)$ be the cost on $[t, t_1]$

if $x(t) = \xi$ and $J(t, \xi)$ the next cost on $[t, t_1]$ if $x(t) = \xi$. Then, the dynamic programming principle states that

$$J(t, \xi) = \inf \left(\int_t^s + J(t, x(s)) \right).$$

The dynamic programming equation (a PDE involving J : the Pontryagin principle was a system of ODEs) is a necessary condition, under some continuity assumptions:

$$\inf_v H(t, \xi, v, \pi) = -\frac{\partial V}{\partial t}.$$

(Those continuity assumptions are not reasonable, but you can get rid of them if you consider viscosity solutions.)

Let $X^{t,x}$ be the solution of

$$\begin{aligned} dX &= \mu du + \sigma dW \\ X_t^{t,x} &= x. \end{aligned}$$

For $f : \mathbf{R}^n \rightarrow \mathbf{R}$, let

$$Af(x) = \lim_{h \rightarrow 0} \frac{E[X_{t+h}^{t,x}] - f(x)}{h}.$$

The operator A is called the generator of the diffusion,

$$A = \mu \frac{\partial}{\partial x} + \frac{1}{2} \text{tr} \left(\sigma \sigma' \frac{\partial^2}{\partial x \partial x'} \right).$$

The expectation $v(t, x) = E[g(X_T^{t,x})]$ satisfies

$$\begin{aligned} \frac{\partial v}{\partial t} + Av &= 0 \\ v(T, \cdot) &= g \end{aligned}$$

(This result can be generalized to $\dot{v} + Av + kv + f = 0$ with initial conditions (Cauchy problem and Feynman-Kac representation) or to $Au - ku + d$ with boundary conditions (Dirichlet problem).) This gives a correspondence between expectations and solutions of PDEs: we can use PDE tools (e.g., finite elements) to estimate expectations or use stochastic tools (e.g., Monte Carlo simulations) to solve PDEs.

Let X be the stochastic process describing the price of an asset, u be a trading strategy (control process, to be determined), $J(t, x, u)$ the expected utility on $[t, T]$ of the wealth resulting from the trading strategy u if the initial price is $X_t = x$, and $V(t, x)$ the expected utility of the optimal strategy. By changing X , we can assume that the cost J only depends on the final value: $J(t, x, u) = E[g(X_T) | X_t = x]$. The control process can be: deterministic (open loop control), Markovian (u_s depends on the current price X_s , but not on previous prices) or adapted to \mathcal{F}_X (feedback control: it can depend on all previous prices). If V is smooth (it is, under restrictive but reasonable assumptions: Lipschitz, bounded),

$$V(t, x) = \sup_u E[V(t + \varepsilon, X_{t+\varepsilon}^u)]$$

(dynamic programming principle, directly applicable in a discrete-time framework). By letting $\varepsilon \rightarrow 0$, we

have the Hamilton-Jacobi-Bellman (HJB) partial differential equation. If V is not smooth, we still have inequalities of the form (for $s > t$)

$$\begin{aligned} V(t, x) &\geq \sup_u E[\liminf V(s)] \\ V(t, x) &\leq \inf_u E[\limsup V(s)] \end{aligned}$$

and would look for viscosity solutions of the PDE.

The optimal stopping problem, *i.e.*, finding a stopping time τ to maximize

$$J(t, x, \tau) = E[g(X_\tau^{t,x})]$$

where X is a diffusion

$$\begin{aligned} dX &= \mu dt + \sigma dW \\ X_t &= x \end{aligned}$$

(not a controlled process: that would be a mixed stochastic control and stopping problem) can also be solved via dynamic programming. Once

$$V(t, x) = \sup_{\tau} J(t, x, \tau)$$

is known, the optimal strategy follows: just check if you are in the **stopping region**

$$\{(t, x) : V(t, x) = g(x)\}$$

or the continuation region

$$\{(t, x) : V(t, x) > g(x)\}.$$

If V is smooth (this is the case under restrictive but reasonable assumptions),

$$V(t, x) = \sup_{\tau} E[\mathbf{1}_{\tau < \theta} g(X_\tau) + \mathbf{1}_{\tau \geq \theta} V(\theta, X_\theta)]$$

for all stopping times θ . (In discrete time, let $\theta = t + 1$.) Using Ito's formula, one can show that V is a solution of

$$\text{Min}\{-(\partial_t + \mathcal{A})V, V - g\} = 0$$

where \mathcal{A} is the infinitesimal generator of the diffusion X ,

$$\mathcal{A}\phi = \mu \cdot D\phi + \frac{1}{2} \text{tr}(\sigma \sigma' D^2 \phi).$$

Control problems can also be solved by verification, but the conditions can be difficult to check; it can be used for the optimal allocation between a risky asset (whose price follows a known but arbitrary diffusion) and a risk-free asset under terminal power utility. A more theoretical example of a verification argument is given to study the asymptotic behaviour of a double stochastic integral $\int_0^t \int_0^u b_v dW_v dW_u$ at $t = 0$.

A super- (resp. sub-)solution of a partial differential equation (PDE) $F = 0$ is a smooth function u such that $F(x, u, Du, D^2u) \geq 0$ (resp. ≤ 0). Whether a \mathcal{C}^2 function is a supersolution of a PDE can be determined by looking at \mathcal{C}^2 test functions:

$$\forall x_0, \phi \quad x_0 = \text{Argmin } u - \phi \implies F(x_0, u, D\phi, d^2\phi) \geq 0.$$

(Rigorously speaking, we need a technical condition: F should be **elliptic**, *i.e.*, non-increasing in D^2u .) The notion of super- or sub-solution can therefore be generalized to non-smooth functions (only the test functions have to be smooth). A **viscosity solution** of a PDE is a (non-necessarily smooth) super- and sub-solution. [Question: how does this differ from distributions and Sobolev solutions?] Some symmetry arguments are no longer valid: $F = 0$ and $-F = 0$ need not have the same solutions. There is a change of variable formula (but you have to replace F by $-F$ if the change of variable is decreasing) and a stability theorem (you can take limits). If $Du \geq 0$, then u is non-decreasing; if $-D^2u \geq 0$, then u is concave (here, ≥ 0 means “is a viscosity solution” – there is no specific notation).

The condition “ x_0 is a minimizer of $u - \phi$ ” can be replaced by “ x_0 is a local minimizer of $u - \phi$ ”, which only involves x_0 , $p = D\phi(x_0) \in \mathbf{R}^d$ and $A = D^2\phi(x_0) \in S_d$ (symmetric matrices). The corresponding sets of (x_0, p, A) triplets are called superjets and subjets, and may lead to simpler proofs. For instance, one can prove the unicity of viscosity solutions (on a bounded open set, if the value on the boundary is given).

The dynamic programming equation (DPE) for the stochastic control and optimal stopping problems remains valid in the viscosity sense when V is not continuous (under some harmless assumptions: continuity).

Fuzzy clustering of short time-series and unevenly distributed sampling points C.S. Möller-Levet et al.

Commonly-used measures of distance between time series (Euclidian distance, correlation, etc.) do not take the temporal order into account. Instead, one can use the Euclidian distance between the slopes of the time series,

$$d(f, g)^2 = \int |f' - g'|^2$$

(this also works for unevenly-spaced time series): this is the **STS distance**.

It can be used by clustering algorithms, e.g., **fuzzy clustering**: find cluster centers (prototypes) v_i and cluster memberships u_{ij} (with $u_{ij} \in [0, 1]$ and $\sum_i u_{ij} = 1$) to minimize

$$\sum_{ij} u_{ij} d(x_j, v_i)^2.$$

The problem is usually solved iteratively: select the v_i at random; estimate $u_{ij}|v_i$; estimate $v_i|u_{ij}$; iterate.

Dynamic orthogonal components for multivariate time series D.S. Matteson and R.S. Tsay (2011)

Multivariate times series modeling studies the time

evolution of μ_t and σ_t :

$$\begin{aligned}\mathcal{F}_t &= \sigma(y_t, y_{t-1}, \dots) \\ y_t &= \mu_t + e_t \\ \mu_t &= E[y_t | \mathcal{F}_{t-1}] \\ \Sigma_t &= \text{Cov}(y_t | \mathcal{F}_{t-1}) = \text{Cov}(e_t | \mathcal{F}_t)\end{aligned}$$

Traditional dimension reduction (PCA), when applied to time series, remove the cross-sectional dependencies, but not the dependence accross time.

To compute dynamic orthogonal components (DOC) in volatility: find a linear transformation of the data $s = Ax$ such that $\text{Cov}(s_t, s_{t-\ell})$ be as diagonal as possible; model the s_i (univariate time seirs) as ARMA processes.

To compute DOC in volatility: remove the trend (e.g., using DOC in mean), find a linear transformation $s = Ax$ such that $\text{Cov}(s_t | \mathcal{F}_{t-1})$ and $\text{Cov}(s_t^2, s_{t-\ell}^2)$ be as diagonal as possible; model the (univariate) time series s_i using a GARCH or stochastic volatility model.

R Journal (2010)

Inverse problems are problems for the form “find x such that $y = f(x)$ ”, *i.e.*, “find the inverse of f ”. If $f : E \rightarrow F$ is linear, injective, then f^{-1} exists, but need not be continuous in infinite dimensions. The Radon transform (tomography) is one such example: it is too sensitive to noise (observation errors). If $f : E \rightarrow F$ is linear with E finite-dimensional and F infinite-dimensional, the problem is overdetermined: some form of *regularisation* is needed. An inverse problem is **well-posed** if the solution exists, is unique, and depends continuously on the problem; if it is not unique or not continuous, it is **ill-posed**. Well-posed problems can be **ill-conditionned**: even if they are continuous, they can amplify observations errors; the *condition number* is the factor by which they amplify those errors.

The following packages can solve differential equations: **deSolve** (initial value problems (IVP), differential algebraic equations (DAE: differential equations plus conservation law), delay differential equations (DDE)), **ReacTrans** (some partial differential equations (PDE): diffusive-advective transport equation), **bvpSolve** (boundary value problems (BVP)), **rootSolve**, **PBSddesolve** (DDE), **sde** (stochastic differential equations (SDE)), **pomp**. (The **odesolve** package is deprecated.)

The **codetools** package provides some functions for code source analysis. Objects have a **source** attribute, that contains their unparsed code (otherwise, we would have to deparse it, and it could end up being slightly different), and a **srcref** attributes that points to the code (it is used by IDEs). They can be used when displaying errors or when setting breakpoints. Check the following functions: **traceback** (call stack after an error), **browser** (explicitly insert a breakpoint), **recover**, **dump.frames**, **sys.calls**, **setBreakpoint** (to set a breakpoint, without explicitly adding a

`browser()` in the code). You may also want to check the `tools` and `utils` packages.

The `hglm` package fits hierarchical generalized models, (e.g., mixed models with heteroscedastic residual variance; Poisson model with gamma random effects; etc.). The interface is still awkward, immature: you have to provide the matrices to describe the model, rather than the model itself; providing the model (say, in the Bugs/Jags syntax) and having the computer *parse* it to extract the matrices would be more user-friendly.

Data cloning is the use of MCMC software for maximum likelihood estimation (the output is not a distribution but a point estimate and a confidence interval). Since MCMC simulations require a prior (when doing bayesian computations by hand, you can remove the prior, which is often just a factor in a formula, but you cannot remove it from MCMC simulations – strictly speaking: replace it by an improper prior), you can reduce its influence by cloning the data $k \gg 1$ times (progressively increase the value). The `dclone` package (with `rjags`, `code`) provides some low-level functions for data cloning.

The `stringr` package provides consistent functions to manipulate strings: names, argument order, behaviour with factors, etc. will no longer be unexpected. All the function names start with `str_`.

The `bayesGARCH` package can fit GARCH(1,1) models with Student T innovations.

Case study: `cudaBayesreg`, bayesian computations in CUDA.

Group testing (aka pool testing) is used when studying low-prevalence diseases: patients are pooled into groups and you only know if someone from the group is affected, *i.e.*, instead of observing $X \in \{0,1\}$, you only observe $\exists i \in [1, n] : X_i = 1$. The `binGroup` package extends the `binom` package to this setup.

A **spike and slab prior** for a Bayesian model is a prior in which each parameter is constrained to be either 0 or in some interval, with only a limited number of non-zero parameters. The `spikeslab` package approximates it with a 3-step process: dimension reduction, model averaging (BMA) and variable selection (elastic regression path, computed with `gnet` – you will also see graphically which parameters are important).

Data compression using dynamic Markov modelling

G.V. Cormack and R.N.S. Horspool
The Computer Journal (1987)

Huffman coding is not optimal: for a binary message and words of size n , it assumes that the probabilities of the words are multiples of 2^{-n} ; in addition, it assumes that the words are independent. This can be improved by increasing the word length, but this also increases the size of the frequency table: some compromise has to be found.

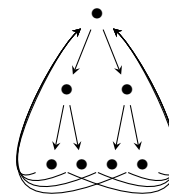
It is easy to fit a Market model to a stream of (bi-

nary) characters, once you know the structure of the Markov chain: just use the sample frequencies, biased to ensure that the probabilities are neither 0 nor 1. Updating the probabilities as we receive the stream of characters gives an *adaptive* algorithm.

To find the structure of the Markov chain (we do not want the best (highest penalized likelihood) one, just a sufficiently good one), start with a simple model, such as



or



and, when a node has enough data, split it. (This is a bit unclear.)

To encode a message knowing the Markov model that generated it, transform it into an *interval* as follow (**Guazzo algorithm**): start with $[0, 1]$; as a new character arrives, split the interval in two proportionally to the probabilities from the Markov model and keep the subinterval corresponding to the character. The bounds of the interval require an increasing precision: it may be helpful to replace the probabilities by approximations (rounding) that do not require that much precision. The compressed message is the final interval, or any number (a simple fraction) in it (you may need the length of the message as well).

Point and figure charting: a computational methodology and trading rule performance in the S&P 500 futures market J.A. Anderson

“Point-and-figure charting” is a technical analysis method method that uses price time series after a change of time: only price jumps above a threshold are included.

Time consistency and moving horizons for risk measures

S.N. Cohen and R.J. Elliott (2009)

Risk measures are often studied for a fixed horizon: you should check how they (and the optimal decision you made using them) change when the horizon changes.

Econophysics: empirical facts and agent-based models

A. Chakraborti et al. (2009)

Contrary to the systems studied by statistical mechanics, systems in social sciences are very far away from equilibrium: this explains why there are no universal laws and why the individuality of economic actors does not completely disappear.

Most statistical properties (stylized facts) of financial data disappear after a change of time (*subordination*) from calendar time to even time (reception of orders), transaction time (execution of order), tick time (price change instants), volume time, return time.

Tick data are *asynchronous* and call for non-trivial variance matrix estimators. The **Fourier estimator** uses the relation between the Fourier coefficients of the returns time series (easy to compute) and those of the coefficients of the variance matrix; the **Hayashi-Yoshida estimator** uses a variant of this idea. Random matrix theory can be used to “denoise” the sample correlation or variance matrix: only keep the eigenvalues that are significantly larger than those you would expect in a random matrix. The distribution of the coefficients of the sample variance matrix; the minimum spanning tree (and its invariants: mean occupation layer (use a large company as the center, or one involved in several sectors), etc.) can also be of interest.

The article ends with a review of agent-based models for the market order book, wealth distribution (and wealth exchanges), herding (minority game and its many variants).

Impact of random failures and attacks on Poisson and power law random networks
C. Magnien et al. (2009)

(Very clear article explaining mean-field approximation to non-physicists; in a nutshell: assume that the (large) graph you have is a tree.)

Real-world networks are often modelled as random networks with a predefined degree distribution (Poisson (exponential decay, *i.e.*, in practice, rates are close to the average) or power law (slower, polynomial decay: larger values are not uncommon)): start with a set of nodes, add edge stubs (according to the chosen distribution), and link them at random. This is a generalisation of the Erdős-Rényi model (limited to the Poisson distribution).

Many theoretical results about random graphs are asymptotic: they are approximations, valid when the graph grows unreasonably large. In this context, whether we are looking at truncated distributions ($X|X \leq k$) or not makes a big difference, especially for power law distributions: if you use the truncated distribution instead of the real one, the asymptotic results you obtain are more likely to be valid for large finite (real-world) graphs as well.

The **mean-field approximation** assumes that the probability that two neighbours of a given node are linked is negligible. In other words, the network locally looks like a tree. The properties of the network can be

examined using **generating functions**, for instance

$$G_0(z) = \sum_k P(X = k)z^k$$

$$G_1(z) = \sum_k P(X = k + 1 | X \geq 1)z^k$$

$$\langle X \rangle = E[X] = G'(0)$$

Since the graph is (sufficiently close to) a random tree, looking at the existence of a *giant component* is akin to studying **branching processes** – hence the omnipresence of generating functions. More precisely, the generating function H of the (asymptotic) distribution of cluster sizes satisfies

$$H(z) = 1 - G_1(z) + zG_1(H(z)).$$

(Generating functions are just a way of encoding distributions and expressing recurrence relations that would otherwise be horribly complicated.) The same equation can help you compute the proportion of nodes to remove to break down the giant component.

The notion of giant component is an asymptotic notion: it is a component that grows linearly with the graph (the mean-field approximation uses a similar but non-equivalent notion: the average size of the components tends to infinity). For finite graphs, you can use the relative size of the largest component: if it is above 5%, you can call it giant.

Experimental results are sometimes in contradiction with the theoretical ones, perhaps because they are just asymptotic, perhaps because we have overlooked some important properties of the graphs – we only looked at the degree distribution, assumed that nothing else mattered, and claimed that the graphs were random. To better model real-world graphs, one can use:

- Constructive models, e.g., preferential attachment (Barabási-Albert);
- Rewired random networks.

Robustness of a graph to failure (random node removal, random edge removal) or targeted attacks (remove nodes with the highest degree, randomly remove nodes of degree at least 2, remove edges between nodes of degree at least 2) can be evaluated using:

- The existence of a giant component (or the proportion of nodes to remove to break it down);
- The average inverse geodesic distance (harmonic mean), which remains meaningful for non-connected graphs.

$$\left(\frac{1}{N(N-1)} \sum_{i \neq j} \frac{1}{d_{ij}} \right)^{-1}$$

Random power law networks are robust to random failures: you have to remove almost all nodes or edges to break down the giant component.

Random graphs with arbitrary degree distribution and their applications
M.E.J. Newman et al.
Physical Review (2001)

For more on the use of generating functions to study random graphs.

Introduction to membrane computing
G. Păun

There are many biology-inspired algorithms (neural networks, evolutionary computing, ant colony optimization, etc.) or computational models (DNA computing). Membrane computing, aka *P*-systems, is another such computational model, based on the remark that in a living body (made of cells, *i.e.*, membranes, containing organelles, *i.e.*, membranes, sending and receiving vesicles, *i.e.*, membranes, contained in organs, *i.e.*, membranes, etc.), almost everything happens on or through a membrane. A membrane computing system is a tree (each node can be thought as a membrane, contained in the parent node, each node contains a set of symbols and rewriting rules (think of \TeX , or random grammars: only one of the applicable rules, selected at random, is applied in each node at each time step), with a few specific symbols to create or destroy membranes, or absorb/secrete symbols from/to neighbouring nodes.

This is just a computational model (not unlike a Turing machine or a quantum computer), with no real world applications.

Lagrange multiplier tests for parameter instability in non-linear models
B.E. Hansen (1990)

To test for structural change, one can:

- Split the sample into, estimate the parameters on each subsample, and compare them;
- Compare (Anova, AIC, etc.) the model with a breakpoint with the model without;
- Idem, but for all possible breakpoints (the breakpoint is rarely known);
- Model the parameter as a random walk and test if its variance is zero;
- Use a Lagrange multiplier test (the estimation problem can be formulated as “minimize the likelihood ratio such that the parameter be constant”): you consider all possible breakpoints but do not have to reestimate the model for each of them.

Jumps and microstructure noise in stock price volatility
R. Sen (2007)

Functional data analysis (FDA) can be used to separate the jump component (microstructure noise) from the realized volatility (*i.e.*, the volatility estimated from high frequency data), after removing the drift by smoothing: FDA is just principal component analysis (PCA) on functions rather than (finite-dimensional) vectors, here, to model the autocovariance structure of the continuous part of the variance process.

Commodity modelling: Schwartz 2-factor model
P. Erb (2009)

Commodity prices can be modelled as:

- A mean-reverting process (Ornstein-Uhlenbeck);
- A geometric brownian motion whose drift is a mean-reverting Ornstein-Uhlenbeck process.

(There is also a 3-factor model, and HJM-type models that also account for the term structure (Hinz), but no details are given).

You can use those models to price options on commodities.

A Few notes on book design
P. Wilson (2009)

This used to be the first part of the `memoir` document class user guide.

Besides the usual advice, the book also provides some font-related vocabulary:

- Serif or sans serif;
- Bracketed serif (smooth transition between the stem and the serif) or unbracketed serif; slab serif (egyptian) fonts can be bracketed or not;
- Vertical or inclined axis (angle of the nibbed pen);
- Small or large counter (loop of some letters: a, e);
- Gradual or abrupt contrast (width difference between thick and thin strokes);

and, less clearly, some font classifications (there are several overlapping or inconsistent classifications, and few actual examples are given):

- Gothic, or blackletter fonts (used by Gutenberg) include the textura (square characters: Goudy Text, Cloister Black), rotunda (round characters: Goudy Thirty) and bastarda (the most common: Fraktur, etc.) families;
- Oldstyle, venitian (Centaur, Berkeley Oldstyle, Jenson);
- Oldstyle, Aldine/French (Bembo, Garamond Palatino, Sabon);
- Oldstyle, Dutch/English (Calson, Janson);
- Transitional, or neoclassical (baskerville, URW Antiqua, Bell);
- Transitional newspaper (better legibility, larger counter, higher x-height, increased contrast) (New Century Schoolbook, Times);
- Modern (exaggerated contrast: Didot, Bodoni);
- Square serif or Victorian (Clarendon, Bero Serif);
- Sans Serif or grotesque or gothic (Helvetica, Futura, Gill Sans);
- Script, Cursive (Brush Script, Zapf Chancery);
- Display or decorative fonts.

R Journal
(June 2011)

It has become difficult to follow the development of all the packages, identify those that are useful, well-

maintained, and reliable (many are just “proof of concept”):

- The task views, cranberries and crantastic tend to be too exhaustive;
- The easiest is to look at the packages most often mentionned on R-help or R-planet (?); you can also check how this changes with time;
- Check the code quality, as measured by the `codetools` package (cyclomatic complexity, coding convention consistency, etc.), from the package metadata itself (licence, number of authors, how many other packages they are maintaining, how active they are on R-help, etc.), or from the version control system (frequency of the releases, number of authors, etc.);
- One could also apply some graph algorithms to the package dependency graph, e.g., looking at the node degree, or applying the page rank algorithm to find the most influencial (used) packages.

The `testthat` package helps you write regression tests as easily as if you were using `stopifnot`, but is more verbose (for instance, you can group the tests, display helpful error messages when they fail, etc.)

Social network analysis (with the `sna` and `tm` packages) can be combined, for instance to analyze mailing lists (thread-author-text triplets)

- Build a network of authors;
- For each word (discard rare words), look at the sub-network of authors using it;
- Build a bipartite graph of authors and words, with edge weights given by the centrality of the authors in the previous graph (the centrality can also help you identify and discard common words);
- Remove the words from the bipartite graph.

Differential evolution (with the `DEoptim` package) is just the traditional algorithm genetic algorithm, but the various operations are no longer combinatorial but modified to be meaningful for real numbers: four instance, “mutation” could be

$$x'_1 = x_1 + F(x_2 - x_3)$$

where x_1, x_2, x_3 are random members of the population. Compared to traditional optimization algorithms (`optim`, `nlminb`), it more efficiently avoids local extrema.

Differential optimization can be applied to portfolio optimization, for instance, when you include the contribution to risk (standard deviation or expected shortfall) in the optimization problem: **risk-parity** portfolio or **equally-weighted risk contribution** (ERC) portfolios (since `DEoptim` does not allow constraints, you can turn them into L^p penalties).

Bayesian model averaging (BMA), which combines several models to yield better forecasts, can be used as a post-processing step and does not need any knowledge of the model details (`ensembleBMA` package); it can be used for weather forecasting.

The `rasterImage` and `grid.raster` function can efficiently plot pixmaps (heatmaps of microarray data, image processing, etc.). You may want to add `interpolate=FALSE`.

Parametrizing correlations: a geometric interpretation F. Papisarda (2006)

Estimators of the correlation matrix need not be correlation matrices at all (e.g., the sample correlation matrix, if there are missing values). Instead, you can parametrize the matrix as $C = BB'$ with

$$b_{ij} = \begin{cases} \cos \theta_{ij} \prod_{k=1}^{j-1} \sin \theta_{ik} & \text{if } j < n \\ \prod_{k=1}^{n-1} \sin \theta_{ik} & \text{if } j = n \end{cases}$$

The n th column of B can be interpreted as the k th basis vector, rotated by angles θ_{kj} , in the (e_j, e_{j+1}) planes.

Statistical properties of world investment networks D.M. Song et al. (2008)

Properties of the network that looks at which country invests in which country (CPIS data from the IMF).

Crossing Intervals of non-Markovian gaussian processes C. Sire (2008)

$\text{Persistence}(\tau, M) = P[\forall t \in [0, \tau] \quad X_t < M]$.

Behavioural and dynamical scenarios for contingent claims valuation in incomplete markets L. Boukas et al. (2009)

In an incomplete market, the no-arbitrage argument does not lead to a unique risk-neutral probability and a unique price for contingent claims: there is a whole interval of arbitrage-free prices. However, the market price is unique: how do we arrive at that unique price?

- Among all the risk-neutral probabilities, take that with the minimum entropy
- Both buyer and seller have some belief (subjective real-world probability) about the future state of the world and are trying to maximize some utility function; by comparing the maximum utility with or without the ability to buy/sell the claim, we can compute a maximum/minimum acceptable price for the claim; if the buyer’s price is higher than the seller’s, a transaction can occur, for instance, with a 1-bid sealed (blind) auction; each party, not knowing the other’s price, will choose a bid that minimizes the maximum **regret** (?); we can imagine that this transaction

- If the buyer's price is lower but both parties are compelled to engage in the transaction, they can enter some kind risk-sharing scheme, minimizing their combined risk, measured as the average (some convex combination) of the drop in utility of each party (but they have to be honest and trust each other);
- If the parties do not have strong beliefs about the future state of the world (real-world probability), but a set of probabilities instead, they can change their beliefs (as little as possible, as measured by the price difference) to come to an agreement (this also requires honesty and trust).

R in Finance 2011

Programming

Functional programming is possible in R, with the `futile.paradigm` package and new operators: `%when%` (for guards), `%must%`, `%isa%`.

Beyond S3 and S4 classes, several packages provide object-oriented facilities: `R.oo`, `proto`, `mutatr`, `futile.paradigm`. While object-oriented programming involves both classes and objects, **prototype programming** (popularized by Javascript) only deals with objects. In R, this can be achieved with *environments*:

```
parent <- new.env()
parent$x <- 1
parent$y <- 3.14
child <- new.env(parent=parent)
child$y <- 2.71
child$x      # Does not work
with(child, x) # 1
sibling <- list2env(
  as.list(child),
  parent=parent.env(child)
)
```

The `proto` package does the same thing, but you do not have to know anything about environments. Possible applications include GUI programming (the code may look simpler than using directly, say, `gWidget`) or the `gsubfn` package.

The `rparallel` package can parallelize computations (on several threads, on a single multicore machine, for embarrassingly parallel problems), with applications to Monte Carlo simulations or boosting.

CUDA-C can speed up SDE simulations (stochastic volatility model) to calibrate the volatility smile (“calibration” means “moment estimator”: you play with the model parameters until something simple computed from the output (a “moment”) matches the same quantity for the real-world data).

The `xtime`, `xts`, `xtdf`, `indexing`, `mmap` packages can help you deal efficiently with time series.

The R ecosystem

Dexy is a tool to generate software documentation and

reproducible research, that claims to be the next generation Sweave: it works with any programming language (R, Python, etc.) or markup format (HTML, L^AT_EX, etc.), is aware of dependencies between datasets and documents, caches intermediate results to avoid having to rerun lengthy computations if nothing changed, lets you reuse the result of the computations in different documents. Contrary to Sweave, it encourages you to separate code and documentation. It can be seen as a combination of Make and Sweave (or templating systems, such as Jinja) or, simply, as a document-oriented Make.

R can run “in the cloud”, thanks to packages such as `Rhive` (Map/Reduce, Hadoop) or `segue` (simplicistic, to run CPU-bound computations on Amazon’s EMR (Elastic Map Reduce)).

R can be used in web applications (from `Rhttpd` (the internal web server, used to display the documentation) or `rApache`) with the `rook` package.

Repast Symphony is a framework to simulate agent-based models (Java, Groovy, Logo); the results can be analyzed in R.

NEOS (Network-enabled Optimization system, an API to send optimization problems over the network, as a web service, in standard formats such as AMPL or GAMS) can be used from R, thanks to the `XML`, `XMLRPC` and `RCurl` packages.

RStudio was mentioned several times.

R can be interfaced with Python (`Pysampler` (?), `PyNum`, `PySci`).

Column stores (e.g., Sybase IQ) are becoming more widespread.

RServe can now be accessed from .NET

R can be used from OneTick, a (commercial) visual data-flow environment.

Finance

R can be used as a quantitative strategy development platform, with the following packages: `quantstrat`, `FinancialInstrument`, `quantmod`, `blotter`, `TTR`, `xts`, `indexes`, `RTAQ`, `signalextraction`, `lspm`, `PortfolioAnalytics`, `PerformanceAnalytics`.

Betfair is a UK betting website, with an API, accessible from R: for instance, you can look at the order book of an event (say, a horse race) and see how it changes with time. Methods from quantitative finance (e.g., algorithmic trading) may be applicable.

Jumps in high frequency data can be identified by looking at the returns/volatility ratio for some robust volatility estimator (e.g., `spotVol` in `RTAQ`); one can also look for jumps in the liquidity, as measured by the order imbalance

$$\frac{\sum \text{Order Sign} \times \text{Order Size}}{\sum \text{Order Size}}$$

or the order book depth

$$\frac{\text{Mean Ask Depth} - \text{Mean Bid Depth}}{\text{Mean Depth}}.$$

The `twitterR`, `RGoogleTrends`, `Infochimps`, `tm`, `XML`, `RCurl` packages can be used to infer the *market sentiment*.

Cap-weighted indices do not perform that well: you may prefer a minimum variance, minimum expected shortfall, or *tangential* (maximum Sharpe ratio) portfolio instead. Try to estimate the returns and the variance matrix robustly: time series factor model (the risk factors are known time series: oil, etc.), cross-sectional factor model (the exposures to the risk factors are known, e.g., industries), statistical models (PCA, ICA). For the minimum expected shortfall portfolio, they use the sample expected shortfall: this is a simple linear optimization problem. The presentation does not explain clearly how they suggest to robustly estimate the returns, needed to compute the tangent portfolio (which quickly disappeared from the discussion).

If two assets X and Y are thought to be related, one can estimate the regression $Y = \alpha + \beta X$ and buy the portfolio $Y - \beta X$. But this **hedge ratio** is asymmetric: instead, replace the least squares regression by the (PCA-based) total least squares (TLS).

Hawkes processes are used to model earthquakes; they account for mainshocks, potentially preceded by (often a single) foreshock and followed by (several) aftershocks, and can also account for dependence between earthquakes in different regions. (They are simply point processes whose intensity depends on previous shocks.) They are strangely similar to volatility time series, and can be used to model financial times series, as the sum of two Hawkes processes, one positive, one negative; the more positive (resp. negative) the process becomes, the larger the intensity of the negative (resp. positive) process becomes (mutually exciting Hawkes processes). One can easily compute the expected realized volatility at a given scale τ and tune the parameters to replicate the observed relation between realized volatility and scale. With two assets, you can also calibrate the relation between realized correlation and scale.

Optimization based on the expected shortfall (ES, aka expected tail loss (ETL) or conditional value at risk (CVaR)) is getting more and more popular for *risk budgeting*. Any homogeneous risk measure (standard deviation, semi-standard deviation, VaR, ES) can be decomposed into a sum of risk contributions (Euler):

$$\text{Risk}(\mathbf{w}) = \mathbf{w} \cdot \frac{\partial \text{Risk}}{\partial \mathbf{w}}.$$

Factor Model Monte Carlo (FMMC) was mentioned several times, with terse details: first fit your factor model (using robust and/or non-parametric methods if possible), use it to simulate data (and backfill missing values), use the simulated data to estimate the distribution of the returns of your portfolio or strategy, use

this (empirical) distribution to compute the risk measures you are looking at. It is not very clear whether they simulate (from an estimated distribution) or resample (*i.e.*, shuffle the data) – both make sense.

The variance matrix of assets returns can be easily estimated even if the assets have a different history, via repeated regressions: start with the assets with the shortest history; for each asset i , estimate the regression $r_i \sim r_{i+1} + \dots + r_n$ (parameters $\hat{\beta}_i$ and variance \hat{v}_i); the variance matrix can be computed, recursively, from the $\hat{\beta}_i$ and \hat{v}_i . If there are not enough data, you can use shrinkage (ridge regression, lasso) or a Bayesian prior, which also provides you with confidence intervals (implementation in the `monomvn` package);

Dependence between Financial Models

D.S. Matteson

R In Finance 2011

The **distance covariance** between two random variables X and Y ,

$$V^2(X, Y) = \|\phi_{X,Y}(t_1, t_2) - \phi_X(t_1)\phi_Y(t_2)\|_w$$

where ϕ is the characteristic function, w a weight function (use t^{d+1} , where X has values in \mathbf{R}^d – X and Y can have different dimensions) is a measure of dependence. It can easily be estimated (just use the empirical characteristic functions).

$$\begin{aligned} a_{ij} &= \|x_i - x_j\| \\ b_{ij} &= \|y_i - y_j\| \\ A_{ji} &= a_{ij} - \bar{a}_{i.} - \bar{a}_{.j} + \bar{a}_{..} \\ B_{ji} &= b_{ij} - \bar{b}_{i.} - \bar{b}_{.j} + \bar{b}_{..} \end{aligned}$$

$$\hat{V}(x, y) = \frac{1}{n^2} \sum_{ij} A_{ij} B_{ij}$$

It depends on the marginal distribution, but you can make it distribution-free by forcing the margins to be uniform (this is sometimes called the *probability integral transform*), *i.e.*, by looking at the copula of (X, Y) .

The distance covariance can be used as a (distribution-free) test of serial dependence

$$H_0 : X_n \perp (X_{n-1}, \dots, X_{n-p})$$

and tell you when to stop adding elements to a time series model.

The distance covariance can be used to define a joint dependence test of (X_1, \dots, X_n) (it is not symmetric):

$$\sum_i \hat{V}(u_i, (u_{i+1}, \dots, u_n)).$$

Minimizing the test statistic leads to an independent component analysis (ICA) – a competitor to the classical FastICA (maximize some (robust) measure of non-gaussianity) and ProDenICA algorithms. This also gives a test for the existence of independent components (the alternative hypothesis of the test is that no matter how you rotate the data, you cannot get rid of the dependence).

Definition of the distance covariance and distance correlation.

Statistical Analysis of financial time series and option pricing with R

S.M. Iacus

R In Finance 2011

Cluster analysis often requires some notion of distance: for time series, one can use the Euclidian distance (it will not give good results but is useful as a benchmark), the correlation, the short time series distance (STS, the Euclidian distance between the slopes of the price time series, $\int |f' - g'|^2$: it also works for irregularly-sampled time series), the dynamic time warp distance (DTW), or the Markov operator distance. The transition operator of a (discrete) stochastic process X is

$$Pf(x) = E[f(X_{n+1})|X_n = x].$$

It is entirely defined by the scalar products

$$\langle P\phi, \psi \rangle = \frac{1}{2} E[\phi(X_n)\psi(X_{n-1}) + \psi(X_n)\phi(X_{n-1})].$$

Evaluated on the functions $1, x, x^2, \dots, x^n$, this gives a matrix: the **Markov operator distance** is the Euclidian distance between such matrices. This can help identify time series with a similar dynamics (e.g., diffusions with a similar drift and volatility), but completely disregards the relation between the innovations driving those time series (most of this talk is about estimating the drift and volatility of a diffusion). This is implemented in the `MODist` function in the `sde` package.

Parameters of financial models, e.g., μ and σ in a geometric brownian motion

$$dX = \mu X dt + \sigma X dW$$

or θ in a CIR model

$$dX = (\theta_1 + \theta_2 X)dt + \theta_3 \sqrt{X}dW$$

(there is a very long list of such models in the presentation) can theoretically be estimated by maximizing the likelihood

$$\prod_{i \geq 0} p(x_{i+1}|x_i)p(x_0).$$

The transition probabilities are rarely known, but approximations are available, leading to Quasi-MLE estimators (the `qmle` function in the `yuima` package).

They can also be estimated with a **2-stage regression** (the shape of the functions f and g will depend on your model):

- Estimate the drift with a regression

$$X_{t+1} - X_t \sim f(X_t)$$

- Estimate the volatility by regressing the squared residuals: $\varepsilon_{t+1}^2 \sim g(X_t)$

- Recompute the first regression, with weights set to the inverse of the variance predicted by the second regression

Model selection often uses the AIC, *i.e.*, the likelihood penalized by the number of parameters in the model. Instead, one can use an L^1 penalty, as in the lasso regression; to choose the coefficients of the L^1 penalties, just use the inverse of the absolute value of the unconstrained parameter estimates. (This is implemented in the `yuima` package.)

Least squares can be used to look for structural changes: estimate the model (e.g., a diffusion) before and after the change point and compute the sum of the squared residuals, for each possible change point; choose the point that minimizes this sum of squares. (The approach presented was different but not very clear: it is limited to breaks in the volatility (the drift is unaffected by the structural change) and does not require the drift to be evaluated). This is implemented in the `cpoint` function in the `sde` package or the `CPoint` function in the `yuima` package.

The `yuima` package can simulate and estimate the stochastic processes most commonly encountered in finance: diffusion,

$$dX = \mu dt + \sigma dW$$

fractional gaussian noise,

$$dX = \mu dt + \sigma dW^H$$

and diffusions with jumps (Levy processes).

Option pricing in R often relies of Rmetrics (`fOptions`, `fAsianOptions`, `fExoticOptions`) or RQuantlib, but is mostly limited to the Black-Scholes model: if you need to go beyond that (jumps, etc.), you have to rely on Monte Carlo simulations. The `yuima` package provides an alternative: **asymptotic expansions**; it allows you to estimate some integral (say, the payoff of the option) of a stochastic process that is “close” to one that is manageable (e.g., deterministic), for instance

$$dX_t = \mu X_t dt + \varepsilon X_t dW_t$$

for ε small.

Modeling microstructure noise with mutually exciting point processes

A. Bacry et al. (2010)

Classical approaches to high frequency data include:

- The **latent price** is a diffusion process, but only a noisy version of it is observed
- Models of the evolution of the whole order book (rather than the price alone)

A difference of two **Hawkes processes** (marked point processes, used to model earthquakes), linked (the more positive the difference becomes, the more the intensity of the negative process increases) can reproduce some stylized facts of high frequency time series:

- Strong macroscopic mean reversion

- Signature plot: the realized volatility on a window of scale τ depends on τ (it is higher at smaller time scales)
- Epps effect: idem for the realized correlation between the returns of two assets (it is lower at smaller time scales)

Sampled at large time scales, those processes look like diffusions.

What is the shape of the Risk-Return relation? **A. Rossi and A. Timmermann (2010)**

One expects the risk-return relation to be monotonic, but the evidence is not compelling.

The question can be settled using boosted regression trees (BRT, regression trees allow for non-linear, non-monotonic relations, and boosting, or more generally ensemble methods, can cope with a large number of variables without overfitting). First, estimate (using BRT) conditional returns and conditional volatility (“conditional” means “using all the information available at that time”, as in “ $E_t[\cdot]$ ”)

$$\text{Stock Return} = f(\text{Risk Factors}) + \text{noise}$$

$$\text{Conditional Return} := f(\text{Risk Factors})$$

$$\text{Conditional Volatility} := \sqrt{\text{Var noise}}$$

(risk factors can include market or sector returns, macroeconomic variables, etc.). Then, still using BRT, estimate the relation between the conditional returns and the conditional volatility: it is not monotonic.

This unintuitive result may be due to the choice of the risk measures. Alternatives include investment opportunities (I wonder how one would measure that) or

$$\text{Cov}(\text{Returns}, \text{Consumption Growth}).$$

Daily consumption growth is not available, but if one replaces it with a **business activity index** (the ADS index combines 10-year and 3-month treasury spreads (daily), initial jobless claims (weekly), employment (monthly), production (monthly), personal income (monthly), sales (monthly), GDP (quarterly) in a dynamic factor model estimated with a Kalman filter), the relation looks monotonic. Such a measure of risk can be seen as “volatility corrected for consumption or business activity”.

Think Positively **K. Pluto and D. Tasche (2005)**

Probabilities of default (PD) can be estimated using the **maximum entropy** principle; to be more conservative, one can look at confidence intervals instead of estimators, or even 1-sided confidence intervals – this approach can give useable results even if there are few or no defaults (low default portfolios), but does not work for small portfolios (or isolated assets).

The article presents a related approach, the **most prudent estimate**. Dependence (between assets, or over time) can be modeled with a Tobit model: defaults

events are triggered by a (component of a multivariate) gaussian random variable exceeding some threshold; for cross-sectional dependence, a 1-factor model is sufficient (?); for intertemporal dependence (the random variables $(X_{it})_t$ can be interpreted as the price, or the “latent value” time series) the correlation suffices: $\text{Cor}(X_{is}, X_{it}) = \rho^{|s-t|}$.

You may want to rescale those estimates, so that the average of the probabilities of default be the probability of default of the whole portfolio: you usually end up with the probability of default of all grades above that of the portfolio.

Twitter mood predicts the market **J. Bollen et al. (2010)**

The vocabulary used in tweets seems to have some predictive power on future DJIA returns: Opinion-Finder uses a positive versus negative lexicon, while Google Profile of Mood States (not public) uses the 6-dimension (calm, alert, sure, vital, kind, happy) (commercial) PoMS lexicon used by psychometricians and extended to include commonly cooccurring terms.

The numeric results look too high.

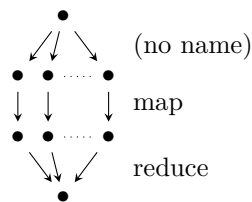
Dynamic Asset Allocation Using Stochastic Programming and Stochastic Dynamic Programming Techniques **Gerd Infanger (2010)**

Dynamic asset allocation, for individual investors, often reduces to “keep the same portfolio $x\%$ stock, $(100 - x)\%$ bonds”, or sometimes “ $(100 - \text{age})\%$ stock”. This is clearly suboptimal. Stochastic programming can provide the optimal strategy for a given utility function. This can be done in two ways:

- Dynamic programming: recursively compute $V(t, w, x)$, the expected final utility of the best strategy if your wealth is w at time t and you invest $x\%$ of it in stocks on $[t, t + 1]$.
- Simulate 10,000 to 100,000 scenarios and transform the stochastic program to a non-stochastic one (the author wrote a software, GAMS/DECIS, to perform this transformation). As often with that kind of problem, the number of constraints is too large (10^{20}), but one can easily identify breached constraints and add them to the problem, until a solution is found.

GraphLab: a new framework for parallel machine learning **Y. Low et al.**

High-level abstractions for graph-based computations, such as Map/Reduce, are too restrictive: they impose the shape of the graph.



Low-level abstractions (MPI, PThreads) are too low-level. GraphLab is in-between: the shape of the graph is arbitrary.

- Contrary to DAG (directed acyclic graph) models, the graph need not be acyclic, and can model iterative algorithms;
- Contrary to the *systolic abstraction*, the decision on which computations should be done next is not made beforehand, but as the algorithm progresses;
- Contrary to MPI/PThreads, the programmer is isolated from synchronization, data race and deadlock problems.

The framework also provides some scheduling facilities (but they could be optimized). The data is stored in the nodes or in the vertices (for data shared by two nodes). Since it is based on PThreads, it is limited to a single machine.

***Pregel: A System
for Large-Scale Graph Processing***
G. Malewicz et al. (Google)
SIGMOD 2010

Pregel is another abstraction for graph-based (?) bulk synchronous parallel computations (BSP): each node (on a separate machine or processor) runs its computations and can send messages to its neighbours, a master keeps track of the progress and of changes in the graph structure (you can add or remove vertices), the computations stop when all nodes tell the master they want to stop. I do not really see the link between graphs and BSP.

***Statistical Approaches
for Network Anomaly Detection***
C. Callegari (2010)

Networks attacks can be detected by:

- Clustering, *i.e.*, unsupervised classification: smaller clusters are attacks;
- Markovian models (e.g., to model TCP states; first-order, time-homogeneous models are simpler and have a better ROC curve);
- Entropy (compress the incoming (log) data with LZW (Lempel-Ziv-Welch), Huffman coding, or dynamic Markov compression, and look at its size)';
- Count-min sketch (use a counting Bloom filter to estimate the average traffic from each IP address and identify deviations from this average, you do not have to store the data separately for each IP address (there are too many): the data is aggregated and distributed);

- Principal component analysis;
- Wavelets (or rather framelets, that provide several time series showing features at different scales; this is very similar to several band-pass filters; that is the most intuitive method presented).

Learning for stochastic dynamic programming
S. Gelly et al. (2009)

Estimating the cost function $V(t, x)$ in a continuous-time dynamic program is untractable in high dimensions, but learning algorithms (neural networks, k -nearest neighbours, kernel regression, regression trees, adaptive discretization, mixture model, etc.) can help.

***On a stochastic knapsack problem
and generalizations***
A.P. Morton and R.K.Wood (1998)

The problem (a knapsack problem where the value of the items is random, the cost function is some expectation or quantile function, but the weight of the items, *i.e.*, the constraints, are deterministic) can be reformulated as a (deterministic) integer program.

***Lectures on stochastic programming:
Modeling and theory***
A. Shapiro et al. (SIAM, 2009)

The notation used to describe stochastic optimization problems (the cost function to optimize is defined as an expectation, the constraints involve quantiles (value at risk), the problem has a time dimension and we are looking for an optimal (feasible) strategy) is very, very heavy. The book deals with the theory underpinning stochastic optimization (duality, etc.), rather than the algorithms that could be used to solve those problems. It assumes that you are already well-versed in classical optimization, for instance, that you understand that duality gives an equivalence between constrained optimization (or regression) and penalized optimization (or regression). The problem can sometimes be recast as a classical optimization problem (e.g., when the random variables have a discrete distribution) but the number of variables explodes. Dynamic programming is a valid approach if the dimension of the problem is sufficiently low (2), but studying the convexity of the quantities to optimize can help overcome the curse of dimensionality.

SIGMOD 2010

Temporality

Databases keep track of the quantiles of the data they contain but rarely keep the history of those quantiles as the database evolves: this could be useful to optimize queries in temporal databases.

Recommendation systems (Amazon, Netflix) could be improved by looking at *temporal patterns* (after buying volume 2 of a series of novels, volume 3 is a good recommendation, but volume 1 is not).

Because of accountability requirements, bitemporal databases (that keep track of all the changes (corrections) that occur on the data) are getting more common (there are almost-standard temporal extensions of SQL and XQuery), but keeping track of *schema changes* (there is a limited number of operations: create, drop, split, merge, copy, join tables; add, remove, rename columns) is trickier: you can keep all tables and columns that ever existed, add a `start_time` and `end_time` to all tables and somehow coalesce the data from the various versions.

Versioned databases ((bi)temporal databases, for text) lead to specific queries, e.g., the durable top- k : the first k elements that were always in the top N in the $[t_1, t_2]$ time interval (with N as small as possible).

Incorrect or imprecise data or results

Sensor data can be turned into *uncertain databases* (aka **probabilistic databases**): instead of storing a value, store a probability distribution (e.g., $U(25, 35)$ instead of 30), which can be multivariate and/or discrete, e.g.,

$$P(\text{foo}, \text{bar}) = .3 \quad P(\text{foo}, \text{baz}) = .6 \quad P(\text{bar}, \text{baz}) = .1.$$

A **probabilistic threshold query** (PTQ) returns all the tuples satisfying the condition with probability above a given threshold.

Approximate evaluation of boolean expressions in (correlated) databases.

Use online algorithms for long-running SPJG (select-project-join-group) queries: users want to see the first results quickly; early result can be approximate

To anonymize time series data (which tend to present temporal correlation), use *Fourier perturbation*: do not add noise to the data itself, but to some coefficients of its discrete Fourier transform.

Relational operations (join, etc.) for incomplete, imprecise or erroneous databases, using mixtures of gaussians to model the data in the tables and choose the confidence intervals

To provide consistent answers from an inconsistent database (resulting from the integration of several databases), use a minimally repaired version of the database and flag the results as “inconsistent” or “consistent”.

To explain why a given tuple is not in the result of a query, you can: represent the query as a tree and identify the node that rejected it; minimally transform the database to include the tuple; minimally transform the query to include the tuple.

When presenting the “top results” of a query to a user, one may want to use a user-specified ordering,

```
SELECT TOP 10 ...
ORDER BY  $\sum_i w_i A_i$ 
```

where the attributes A_i are part of the data, and the weights w_i defining the preference function are given by the user. If the user is unsure of his weights, we

can display the **skyline** (efficient frontier) results, *i.e.*, those that are at the top, for some choice of weights (actually, this is *not* equivalent) and try to infer the weights from the skyline results the user prefers. The resulting weights, for one or several choices, can be displayed on a *parallel plot*.

Non-tabular data types (graphs), new types of queries

To search for a graph or a subgraph in a graph database, look at frequent and infrequent *graph fragments*. The same idea applies to supervised classification of graphs (e.g., to distinguish program flow graphs with and without bugs): look for discriminative subgraphs, *i.e.*, subgraphs frequent in one set but not in the other.

The *frequent itemset* problem can be extended to graphs with a bag of labels on their vertices: you then look for itemsets that appear on nearby nodes. For each node and each label, look at the number of neighbours at distance k with that label; this can be approximated by a Markov model (influence propagation, film recommendations in a social network, etc.).

Unstructured or textual data is often explored through k -nearest neighbour (k -NN) queries.

Knowledge is increasingly represented as RDF statements (aka subject-verb-object (or subject-attribute-value) sentences, or directed graphs with coloured edges), e.g., DBpedia.

Statistics

Hidden Markov models can be used to infer the internal state of a system (intrusion detection using log files).

You can identify outliers in relational data as follows: turn the foreign key constraints into a **graphical model** (*relational dependency network* (RDN)); approximate the conditional probability distributions of the nodes, e.g., using regressions (this may automatically identify missing constraints); use the resulting confidence intervals to identify outliers and/or clean the data.

Optimizations

Very long boolean expressions can be evaluated efficiently, without having to convert them to disjunctive/conjunctive normal form (DNF/CNF, “normalized” means that the alternating AND/OR tree has depth at most 2), which requires exponential space: you can lazily convert the expression to a normal form; you can encode the boolean tree as a list of *Dewey ids* (number the children of each node from 1 to n , use a special symbol $*$ for the last child of an AND node, and process them in order); you can encode the boolean tree as a set of intervals, the expression begin satisfied if those intervals cover $[0, 1]$ (use $[0, 1]$ for the root, the children of an OR node have the same interval as their parent, the intervals of the children of an AND node for a partition of the interval of their parent), for the implementation, discretize the intervals (replace $[0, 1]$

by $\{1, 2, 3, 4, \dots, n\}$). Those algorithms are used by *advertising exchanges*, in which websites, advertisers and intermediaries form a graph, with boolean conditions on the edges describing the desired profile of the clients (they could be static or automatically generated and modified to optimize the number of clicks) and one wants to evaluate the expressions on all the paths (not just the edges) between a website and all advertisers.

Most databases process queries independently of one another: exploiting dependencies between queries (from different users, or subqueries of a single large query involving many aggregations) can speed up data retrieval.

Computing the correlation of N time series of size T can be done faster than $\Omega(TN^2)$.

Some people write Ajax-producing SQL queries.

***Distilling free-form natural laws
from experimental data***
**M. Schmidt and H. Lipson
Science (2009)**

Symbolic regression is an evolutionary algorithm that searches a space of mathematical expressions, less constrained than linear or non-linear regression; instead of producing a single solution, it can provide a set of solutions on the accuracy–parsimony Pareto front. It can be used to identify *invariants* (conservation laws) in physical systems.

Trivial or approximately trivial invariants ($\cos^2 \theta + \sin^2 \theta$, $17 + 1/(1 + x^2)/100$) can be discarded by looking at how predictive the conservation law is:

- Given \mathbf{y} and a conservation law $f(x, \mathbf{y}) = 0$, you should be able to find x ;
- The conservation law $f(\mathbf{x}) = 0$ produces other conservation laws $\partial f / \partial x_i = 0$, which should be predictive as well.

The laws you find depend on the data provided to the algorithm:

- For instance, for a physical system, with only positions, you will have the description of the manifold in state space; with velocities, you will have energy laws; with acceleration, you will have the equations of motion;
- With restricted datasets, you will have approximations (e.g., if you only have small angles, the conservation laws will replace $\sin \theta$ by θ);
- With a limited set of building blocks, you will have approximations (e.g., $\sin \theta \approx \theta - \theta^3/3!$, if there is no sine function) or equivalent formulations ($\sin \theta = \cos(\theta - \pi/2)$ if you have cosine but not sine);
- With several datasets, you can have more general laws (e.g., with datasets corresponding to different masses, you can identify the role the mass plays).

The algorithm can be made aware of units.

Knowledge discovery can be made incremental:

- You can seed the algorithm with known formulas

(e.g., the conservation laws for a double pendulum are likely to contain quantities that look like those for a single pendulum);

- Looking at subexpression frequencies across different systems can help identify meaningful quantities (momentum, potential energy, etc.);
- Since many formulas contain repeated terms, the laws are not coded as an expression tree (it would have to have identical subtrees, which is not very parsimonious), but as a list of assignments of the form $x_{i+1} \leftarrow f(x_1, \dots, x_i)$.

The time to convergence is exponential in the complexity of the formula and quadratic in the number of variables.

***Can a biologist fix a radio?
or, What I learned studying apoptosis***
Y. Lazebnik (2002)

To understand the stagnation of research in biology (thousands of articles published every year on very narrow subjects (often, a single protein or gene), failing to lead to new drugs or a better understanding of biological phenomena), the author suggests to imagine how a biologist would try to study and fix a radio (open it, describe it, try to manually remove elements, randomly destroy some elements (by shooting the radio), etc.). Biologists lack a *quantitative* language, with which they could make predictions; they lack a *systems approach* (when graphically representing the subject of their research, they put it at the center of a diagram and come to the conclusion that it is related to everything else).

Can a systems biologist fix a tamagochi?
L. Cardelli (2007)

(Epistemology in the information age)

The previous article focused on hardware reverse engineering (a radio), but does not work with biological organisms, which are more like software (the article uses the tamagochi as an example):

- Principle approach: try to find the principles that govern the organism: creator, documentation, function, etc.;
- Mechanistic approach: identify the parts, check how they are connected, how they react to perturbations;
- Behavioral approach: reactions to stimuli, in a population;
- Environmental approach: how the organism evolved; behaviour in and interactions with its environment.

(Biological analogues of) software reverse engineering tools may be useful: tracing, breakpoints, core-dumps, stackdumps, packet sniffing, reverse compilation, power analysis, etc.

Categories for the practicing physicist
B. Coecke and E.O. Paquette

***The R Inferno*
P. Burns (2009)**

Very long list of common problems and caveats for (not only) beginning R users.

***From the fundamental theorem of algebra
to astrophysics: a “harmonious path”*
D. Khavinson and G. Neumann
Notices of the AMS (55)**

The effect of gravitational lenses can be described by polynomials in z and \bar{z} (of the form $h(x + iy) = P(z) + Q(\bar{z})$, $P, Q \in \mathbb{C}[X]$; they are actually harmonic, *i.e.*, $\Delta h = 0$). The number of images is the number of roots. It is trickier than with (analytic) polynomials: for instance, $h(z) = z^n - \bar{x}^n$ has an infinite number of roots.

***Missing data: our view of the state of the art*
J.L. Schafer and J.W. Graham
Psychological methods (2002)**

To deal with missing data, avoid old methods (case deletion, simple imputation) and prefer:

- Maximum likelihood, *i.e.*, including the missingness of the data in the model; just discarding the factors (in the formula for the maximum likelihood) involving missing data is not a good idea; the estimation can be performed with the EM algorithm;
- Bayesian multiple imputation: replace the missing values with an estimate of their distribution (not a single value, as with simple imputation), or a set of values, yielding several plausible complete datasets – surprisingly few such surrogate datasets are sufficient (3 to 10).

You should be aware of the distinction between:

- Data missing completely at random (MCAR): missingness does not depend on X or Y ;
- Data missing at random (MAR): missingness does not depend on X , but may depend in Y ;
- Data missing not at random (MNAR): missingness depends on Y .

Also check the Multiple imputation FAQ.

***Hash visualization: a new technique to
improve real-world security*
A. Perrig and D. Song**

Humans are bad at comparing hash strings: instead of strings, one can use images, generated by an expression tree, built from the hash string and a (stochastic) grammar (use the hash string to choose which branch of the grammar you should take). This is not unlike L-systems.

***Continuity analysis of programs*
S. Chaudhuri et al.
POPL 2010**

It is not too difficult to decide whether a mathematical expression represents a continuous function (formal algebra systems should be able to simplify most expressions of the form $f(x) - \lim_{h \rightarrow 0^+} f(x + h)$); for imperative programs, it is trickier, but can be done (in most cases).

***Mathematics and the internet: a source of
enormous confusion and great potential*
W. Willinger et al.
Notices of the AMS (56)**

The popular claim that “the internet is a scale-free network; it is robust to random attacks but very vulnerable to targeted attacks” may be incorrect:

- It is based on `traceroute` data, that ignores the fact that routers have multiple interfaces and cannot see the low-level infrastructure (below the IP layer – this is used, invisibly, to make the network faster and more robust);
- It assumes that the network is random: this is not true, since the addition of a small number of edges can greatly increase robustness – ISPs are probably already doing that, to increase their quality of service (QoS);
- Vulnerabilities are at the edge of the network: if an ISP disappears, all its clients are disconnected.

***Genetic optimization using derivatives:
the `rgenoud` package for R*
W.R. Mebane and J.S. Sekhon**

Genetic optimization algorithms can be improved by modifying each candidate solution by a local search, so that they all be local extrema.

***Gestion d’arbres
par représentation intervallaire*
SQLPro (2003)**

Trees, in SQL, can be efficiently represented by intervals:

- Leaves are 1-element intervals, (from 1 to n);
- A node is represented by the set of leaves it leads to;
- The order on the leaves is chosen so that each node is represented by an interval; in particular, leaves with the same parent are adjacent.

This is very efficient if the tree is static, but insertion is $O(n)$ (you have to shift half the tree on average).

***Music: broken symmetry,
geometry and complexity*
Gary W. Don et al.
Notices of the AMS (57)**

Gabor transforms (aka spectrogram, sonogram, short-time Fourier transform: the FFT on a sliding (smooth) window of constant size) has musical applications, such as:

- Help singers analyze and improve their vibrato;
- Analyze reverberation;

- Show (quantify) beatings between overtones;
- Decompose a piece into sections (and perhaps also identify similarities between sections);
- Show the structure of those sections (in a more graphical, directly understandable way than traditional notation);
- Transcribe birdsong (zoomusicology);
- Analyze acoustic phenomena, such as the Shepard tone (it does appear in Renaissance music, as voice painting for an infinite upward movement).

The Gabor transform can be seen as the dot products with a set of vectors, the **Gabor frame** (it is not a basis because the windows overlap); the generalized inverse (the dual Gabor frame) can be used to define filters (amplify some region of the spectrogram, remove noise, slow down without changing the pitch, change the pitch, etc.) or for synthesis (draw the spectrogram and transform it into sound – it is a form of *granular synthesis*, the elements of the Gabor frame being the grains).

The **scalogram** (continuous wavelet transform, CWT) is similar but lets the window size vary.

The **percussion scalogram** is obtained as follows: filter the signal (e.g., only retain frequencies in a given band, to isolate an instrument); compute its spectrogram; build a pulse train (a 0-1 signal) by comparing the intensity of the spectrum with a given threshold; take the scalogram of the pulse train. This can help compare the rhythm of various instruments.

The **entropy** of the intervals between two percussive strikes is a measure of the complexity of the rhythm. However, this discards time: you may prefer the **Markov-1 entropy** (average of the entropy of the transition probabilities) (?).

There is no mention of the **cepstrum**, more appropriate to study pitch classes (rather than pitches), or the **Mel-frequency cepstrum** (MFC), used in speech recognition.

Time-frequency analysis of musical rhythm X. Cheng et al. (2010)

More details on the percussion scalogram, and examples of combined analysis of rhythm and melody, using both the spectrogram and the percussion scalogram:

- It can distinguish between strong and weak beats: there is no amplitude information in the pulse train, but strong beats happen to be longer;
- The Haar CWT may look more appropriate to study a pulse train, but it leads to less readable a picture, with “cubist” artefacts;
- The article ends with some advice on the choice of parameters.

The percussion scalogram is only a first step towards a better graphical notation or representation of the hierarchical rhythmic structure of a piece – the article uses (ugly) ascii art to mimick the already inadequate traditional notation, only replacing the hierarchi-

cal cues (measures, slurs) with parentheses. The task looks similar to the (automated) genome annotation problem in biology.

Music and mathematics T. Fiore

Music can (sometimes) be analyzed as combinatorics or group actions on the following sets:

- Pitch classes: $\mathbf{Z}/12$ or $\mathbf{Z}/7$;
- Pitch class sets: $(\mathbf{Z}/12)^n/\mathfrak{S}_n$;
- Chords: $(\mathbf{Z}/12)^3$.

You can build group actions from transposition (translation) and inversion

$$(a, a + b, a + b + c) \mapsto (a, a - b, a - b - c);$$

this defines an action of the dihedral group D_{24} on $(\mathbf{Z}/12)^3$.

The PLR group acts on the set S of major and minor chords (the orbits of $(0, 4, 7)$ and $(0, 3, 7)$ under the action of $\mathbf{Z}/12$ acting by addition) and is generated by the following transformations: P (parallel major or minor, e.g., $C_{maj} \mapsto C_{min}$), R (relative major or minor, e.g., $C_{maj} \mapsto A_{min}$), L (leading tone change, e.g., $C_{maj} \mapsto E_{min}$). This is another action of the dihedral group D_{24} . Chord progressions are paths in S , and often follow elements of the PLR group.

The formulas do not seem to make sense.

$$\begin{aligned} P(a, b, c) &= (c, \cdot, a) \\ L(a, b, c) &= (c, \cdot, b) \\ R(a, b, c) &= (b, a, \cdot) \end{aligned}$$

Musical actions of dihedral groups A.S. Crans et al. (2007)

More details on the two actions of D_{24} on $(\mathbf{Z}/12)^3$.

The topos of triads T. Noll (2005)

Instead of group actions on $\mathbf{Z}/12$, you can consider monoid actions. For instance, the affine transformations $z \mapsto az + b$ on $(\mathbf{Z}/12)^3/\mathfrak{S}_3$ that preserve the major triad $(0, 1, 4)$ (the author encodes pitch classes with the circle of fifths: $0=C$, $1=G$, $2=D$, etc.) for an 8-element monoid T . The topos of triads is \mathbf{Set}^T ; some of its properties (subobject classifier, etc.) have musical interpretations.

[The math review is more readable than the article.]

Visual hierarchical key analysis C.S. Sapp (2005)

To graphically represent modulation and harmony changes in a piece of music, plot it in a triangle, with the colour of point (x, y) representing the key (from

the Krusmansl-Schmuckler key-finding algorithm) of the subset of the piece of length $1 - y$ centered on x (where $y \in [0, 1]$, $2x \in [1 - y, 1 + y]$, and the total length of the piece is 1).

***Midi Toolbox: Matlab tools for music*
T. Eerola and P. Toivainen (2004)**

Once your mathematical software can read, play, write Midi files, you can also:

- Plot a piece as a piano roll,
- Look at the distribution of pitch classes or intervals,
- Look at the pitch class transition matrix,
- Look at the autocorrelation of the melodic contour,
- Apply key-finding algorithms: the Krusmansl-Schmuckler algorithm just compares, by looking at the correlation, the distribution of pitch classes with corpus-based references for major (♭♭♭♭♭♭) and minor (♭♭♭♭♭ – proportions of C, C#, D, etc.); there are also SOM-based algorithms (self-organizing maps) and meter-finding algorithms (look at the autocorrelation of the note-onset time series);
- Compute the melodic complexity (entropy),
- Compare pieces in a corpus, etc.

***Statistical methods for corpus exploitation*
M. Baroni and S. Evert (2006)**

Introduction to statistics (tests, estimators, etc.) for linguists.

In R, use `binom.test` for tests about proportions (e.g., the proportion of passive sentences in English).

For a longer introduction to statistics (anova, tests), check *The Foundations of Statistics: A Simulation-based Approach*, by S. Vasisht and M. Broe (Springer, 2010).

EuroSciPy 2010

Scientific computing in Python relies on SciPy; specific bundles of modules, targeted at a given domain, are packaged into “SciKits”.

***Supply and demand shifts
in the shorting market*
L. Cohen et al. (2006)**

Comparison of short supply, short demand and future returns: in markets with bad information flow, there is a relation, suggesting that short selling plays a role in price discovery.

***Asset prices under short-sale constraints*
Y. Bai et al. (2006)**

Short sale constraints have contradictory effects on the markets:

- It limits the ability to share risk, limits the supply of stocks, and leads to an increase in price;
- It limits how much information flows into the prices (some informed investors cannot invest as much as

they would like to), increases uncertainty and volatility, drives investors away and decreases prices (as usual: opacity in financial markets leads to price drops and crashes).

The authors use a model of the market with two periods, two investors, with different endowments and utilities.

***The large scale structure
of semantic networks: statistical analyses
and a model of semantic growth*
M. Steyvers and J.B. Tenenbaum (2005)**

Natural language semantic networks (WordNet, Roget’s thesaurus, word associations) are not completely unstructured: rather than a rigid structure, they have some statistical properties (small world, scale-free, etc.) that can be explained by a preferential attachment model.

***25 years of IIF time series forecasting: a
selective review*
J.G. De Gooijer and R.J. Hyndman (2005)**

Review of recent and not-so-recent progress in time series forecasting: state space models (exponential smoothing, Kalman filter, structural models aka dynamic linear models), ARIMA (VAR, ECM), seasonality, non-linear models (threshold, regime switching, neural nets), long memory (ARFIMA: $0 < d < \frac{1}{2}$), GARCH (GARCH(1,1) if often sufficient), evaluation of the quality of a forecast, combining forecasts (simple average (robust), weighted average with weights from OLS or OLS with a $\sum w_i = 1$ constraint, time-changing weights), prediction intervals or densities (bootstrap) (do not forget to account for power uncertainty).

There has been little progress on: count data forecasting, panel data.

***Domain-specific languages:
an introductory example*
M. Fowler (2007)**

Java is too verbose to implement simply a state machine: XML can help, low-overhead markup languages (YAML, JSON) are better, but DSL (domain-specific languages), which add some simple control structures, are even better – M4 was an early example. A “program” in a DSL is often simpler than a program in a general language, and can be easily represented graphically (cf. graphical programming languages: PureData, Scratch/Etoys/Squeak).

***Economics needs a scientific revolution*
J.P. Bouchaud (2008)**

Economics is based on very strong assumptions (axioms), that prevail on empirical evidence: the marketplace has been deified. These dogmas are perpetuated through the education system: we need more natural

sciences in the curriculum. For instance, models based on those incorrect axioms actually assume that there will be no crisis. markets are wild: they are not in a state of equilibrium and will never be. Financial innovations should be tested by independent agencies, as for all potentially dangerous industries.

***Trust! Why it has been lost
and how to regain it*
D. Sornette (2008)**

The recent crises (ITC, housing, MDS), each feeding the next, are the result of a loss of trust and a lack of governance, rather than incorrect mathematical formulas. They come from *endogenous* instabilities (speculative euphoria) rather than exogenous factors: they are therefore predictable. Crises are often beneficial: they create an excess capacity that has a long-term impact on new technologies (rail in the UK just before the industrial revolution, the IT bubble, etc.). Market movements, business cycles, unexpected downturns will not disappear: they are part of a healthy economic system.

Regulations are needed, but they are often too simple or too complex, and have unintended consequences – they give an illusion of control to regulators. We need to develop a “culture of risk” (I would call it “risk literacy”) for managers in governments, regulatory bodies, financial institutions – and also in the general public.

***Dynamics of market correlations:
taxonomy and portfolio analysis*
J.P. Onnela et al. (2003)**

The minimum spanning tree built from the correlation matrix of daily asset returns is unstable: to see if it is significantly different from one period to the next, we can look at numerical properties of the graph (akin to topological invariants, in algebraic topology), such as its degree distribution (or just the tail of the distribution, if the graph is scale-free: $f(n) \sim n^{-\alpha}$)

or the average distance to the “central” vertex (e.g., that with the highest degree, the highest weighted degree (use the correlation coefficients as weights) or that giving the lowest average distance). One can also look at the proportion of edges present in the tree on two consecutive months. (The authors use a 4-year window: crises will create a discontinuity twice, when they enter and leave the window.)

The branches of the MST may be similar to some sector classifications.

The minimum portfolio (based on the sample covariance matrix, *i.e.*, the same matrix used to build the minimum spanning tree), mainly contains leaves of the MST; the maximum return portfolio is still far from the center, but not as much. This just says (graphically) that optimal portfolios are diversified (far from the center).

***The virtues and vices of equilibrium
and the future of financial economics*
J.D. Farmer and J. Geanakoplos (2008)**

Equilibrium theories usually arise from game theory: models with several agents (producers and buyers), each with some initial goods (*endowment*), some ability to transform those goods (*technology*), and some convex *utility*; they claim that there is a supply-demand (Arrow-Debreu) equilibrium. But this equilibrium may be unnatural or problematic: at equilibrium, supply magically matches demand: at the end of the day, there is no unsold inventory; the equilibrium need not be stable or even unique; there could be some attractor (the economy is a dynamical system) preventing you from reaching the equilibrium; it may change with time; the convex utility is contradicted by behavioural economics.

There are two justifications for the economy being in an equilibrium: the *tatonnement* mechanism (economic agents progressively adjust their consumption and production and approach the equilibrium) and *omniscience* (if everyone knows the equilibrium and everyone knows that everyone knows about it (and knows that everyone knows, etc. – common knowledge), it is in everyone’s interest to use the equilibrium prices).

The economic equilibrium theory can be extended into a *financial equilibrium theory*, by adding time, uncertainty and financial securities; because of uncertainty, utility is often replaced by *expected utility*.

Equilibrium often entails *efficiency* (in complete markets, *i.e.*, if it is always possible to offset any future risk): Pareto efficiency (there is no change in prices that would benefit everyone); information efficiency (prices are unpredictable, e.g., are martingales, *i.e.*, all the information available is already included in the prices; this implies that the price of assets coincide with their *fundamental value* – the present value if there is no uncertainty); arbitrage efficiency (absence of arbitrage).

Contrary to the large number of macronomic empirical laws that stress mere correlations, equilibrium theories (based on an agent model) can produce cause-consequence relations: they are useable for policy decisions. They do not incorporate psychological biases and the bounded rationality of economic agents: this is unrealistic, but parsimonious and hopefully sufficiently good.

Even if equilibrium models are wrong, they are useful: one can use them to find and exploit deviations from equilibrium; one can assume that there is no arbitrage in one market (e.g., interest rates) to help exploit arbitrage opportunities in another market.

Equilibrium theories emphasize understanding over predictability, and often make auxiliary assumptions: they are (almost) not falsifiable. The presence of *power laws* is often advanced as an argument against equilibrium, because in physics, power laws appear in phase transitions and never in an equilibrium – but they are

not incompatible with the notion of equilibrium. The progress towards equilibrium is unnaturally slow (de-cause instead of years, if one only has daily market data).

The causes of the recent crisis are manifold: structural changes (regulatory changes supposed to protect people with a bad credit record), statistical mistakes (looking at the mean instead of the whole distribution), perverse incentives (rating agencies), people trying to hide what was happening. Even if they are wrong, equilibrium theories and game theory can be useful (e.g., to study perturbations around equilibrium) but should be combined with behavioural and experimental finance and more structural models (e.g., how does the continuous double auction mechanism (*i.e.*, the use of limit orders and markets orders, as in most financial markets) influence prices?). Biology can provide some inspiration as well: for instance, one could study the taxonomy and evolution of financial strategies, or the market environment (investors, brokers, regulators, banks, governments, etc.). Financial markets are a complex system.

R in Finance 2010

Indirect inference suggests to use an auxiliary model, farther away from the data, but easier to fit.

R can interact with more and more third party systems: iBrokers, OneTick, etc.

Repast Symphony is a (free, Java) tool to simulate **agent-based models**; the results can then be analyzed in R.

Esper is an SQL-like processing language for asynchronous data; the callback functions run when an event is received can be written in R; it can use Redis as a key-value store.

R can provide a complete toolchain for fund managers: data (`quantmod`, `indexes`, `RTAQ`, `xts`), simple computations on data (`TTR` (technical analysis), `signalextraction`), trade selection (`quantmod`, `quantstrat`), risk control (`PortfolioAnalytics`, `lspm`), backtest or performance monitoring (`blotter`, `FinancialInstrument`, `PerformanceAnalytics`), etc.

The `indexing` package provides direct access (using the `[]` operator) to vectors stored on disk and mapped to memory on demand (MMAP); an index is kept in memory to speed up data access.

The following packages were also mentioned: `signalextraction`, `ghyp` (for non-gaussian distributions), `Rhive` (Map/Reduce with Hadoop, on Amazon's EC2 cluster), `tm` (text mining, the computations can be distributed with Hadoop/Hive).

GPUs can be used for high-performance programming.

Les nouvelles formes d'organisation du travail Xavier de la Vega (SH, 2010)

Company structures can be classified into one of the

following four types, which can be expressed in terms of graph theory:

- Clique (unstructured company, often very small);
- Trees (strong vertical structure: taylorism, pioneered by Ford, dominant in southern Europe and emerging countries);
- Graphs full of small clusters or cliques and with a small diameter and average distance (horizontal structure: lean production, pioneered by Toyota, dominant in the UK and the US, combining low inventory (just-in-time production), diversity of the products/tasks (a given employee can perform several tasks, so as to better adapt to changes in demand), and quality; it became more widespread with the advent of computers);
- Dynamic networks, *i.e.*, graphs changing over time (learning organizations, dominant in Northern Europe, based on autonomy (you are given objectives, but are free to choose how to achieve them) and communication between workers).

Multiple testing corrections Silicon Genetics (2003)

When performing multiple statistical tests (often, tens of thousands), p -values have to be corrected, for instance with the following methods:

- Bonferroni correction: $p^* = \text{Min}(1, np)$;
- Bonferroni step-down (Holm) correction,

$$p^* = \text{Min}((n - k + 1)p, 1),$$

where k is the rank of the p -value (sort them in ascending order);

- Westfall and Young correction: resample the data to have an estimate of the distribution of the sorted p -values under the null hypothesis, and consider the sorted p -values significant as long as they are under their resampled estimates;
- Benjamini and Hochberg correction:

$$p^* = \frac{n}{n - k} p.$$

The first 3 methods control the familywise error rate (FWER), while the last controls the false discovery rate (FDR).

Maximum entropy distribution inferred from option portfolios on an asset C. Neri and L. Schneider (2009)

Maximum entropy estimators make no distributional assumptions: given a set of option prices for the same underlier and maturity, one can compute the distribution of prices at expiry that maximizes entropy and agrees with the option prices. The article provides detailed computations for calls and digitals, with theoretical justifications.

***Social effects in science: modelling agents
for a better scientific practice***
A.C.R. Martin (2009)

Scientific discovery can be seen as an *opinion dynamics* model on a social network, endowed with two error-correcting mechanisms:

- Experiments, *i.e.*, interaction with nature, seen as an external field;
- Retirement of old scientists.

The bivariate normal copula
C. Meyer (2009)

20 pages of formulas.

Estimation of the instantaneous volatility
A. Alvarez et al. (2010)

Study of the speed of the convergence of

$$\varepsilon^{1-p/2} \sum_i |X_{i\varepsilon} - X_{(i-1)\varepsilon}|^p \longrightarrow m_p \int_0^t \sigma_s^p ds$$

where $dX_t = a_t dt + \sigma_t dW_t$ is a stochastic volatility model (σ_t is a càdlàg martingale) and m_p a constant,

$$m_p = E |Z|^p, \quad Z \sim N(0, 1).$$

***Most efficient homogeneous
volatility estimator***
A. Saichev et al. (2009)

Common estimators of volatility based on OHLC prices, such as

$$\hat{\sigma}_{RS} = \sqrt{\frac{(H - O)(H - C) + (L - O)(L - C)}{T}}$$

$$\hat{\sigma}_{GK} = \sqrt{\frac{k_1(H - L)^2 - k_2((C - O)(H + L) - 2(H - O)(L - O)) - k_3(C - O)^2}{T}}$$

(for some magic values of k_1, k_2, k_3) are not the most efficient. This article studies quadratic estimators (*i.e.*, quadratic forms on $H - O, L - O, C - O$) or more generally homogeneous estimators, and identifies the best one, depending on the model describing the underlying stochastic process.

***Impact of the tick size
on financial returns and correlations***
M.C. Münnix et al. (2010)

The distribution of stock returns (e.g., tail behaviour), the correlation between stock returns, are known to depend on the horizon in a non-obvious way. This could be due to rounding, *i.e.*, to the tick size (either imposed by the exchange, or through psychological reasons leading to clusters at some multiples of the tick size – the *effective tick size*) and can be compendated for.

***A new composite index
of coincident economic indicators in Japan***
S. Fukuda and T. Onodera (2001)

Coincident indicators (that give the state of the economy, *i.e.*, the position in the business cycle) such as industrial production, sales, employment, income (income is used in the US but not in Japan) are naively combined (average growth rate) by the Japanese Economic Planning Agency to produce a “composite index (CI) of coincident indicators”. It is biased towards industrial production, can be inconsistent with leading indicators (that try to predict the state of the economy in the coming year) or even the economic situation. It can be replaced by a less naive model, assuming that all those indicators come from a single hidden factor, that can be estimated with a Kalman filter (Stock-Watson’s “single index dynamic factor model”).

***Multiscaled cross-correlation dynamics
in financial time series***
T. Colon et al. (2010)

Correlation matrices of stock returns have been studied in the following situations:

- Whole spectrum of the sample correlation matrix, estimated on a fixed period of time;
- Evolution, with time, of the largest eigenvalue of the correlation matrix estimated on a moving window of fixed size.

This article suggests to:

- Look at the whole spectrum, not just the first eigenvalue;
- Look at the dependence on both window position and window size, as in wavelet analysis, or look at the matrix of **wavelet correlations** instead: the correlation between the wavelet coefficients at a given location and scale.

***The effect of discrete vs continuous-valued
ratings on reputation and ranking algorithms***
M. Medo and J.R. Wakeling (2010)

Iterative refinements of user and object reputation (à la PageRank) give more accurate ratings than naive averages, but restricting the ratings to a discrete set of values (e.g., a 5-star system) significantly degrades the performance of the algorithm.

***Using Permia and Nsp for constructing
a risk management benchmark
for testing parallel architecture***
J.-P. Chancelier et al. (2009)

Premia is a non-free library developed by the Inria to price derivatives. Nsp is a free Matlab-like language developed by the École des Ponts, not unlike Octave, Scilab or R. They also use MPI for parallelism.

Recurrence networks – a novel paradigm for non-linear time series analysis
R.V. Donner et al. (2009)

There are many ways of constructing a graph (or network) from a time series (or a (single) trajectory of a dynamical system):

- The **recurrence network**, whose adjacency matrix is given by the recurrence plot,

$$R_{ij}(\varepsilon) = \mathbf{1}_{\|x_i - x_j\| \leq \varepsilon}$$

(you may want to remove self-loops; as you change ε , you actually have a family of networks);

- The visibility graph (whose vertices are points in time, and there is an edge between t_1 and t_2 if the $[x_{t_1}, x_{t_2}]$ segment remains above the curve);
- The transition matrix (a weighted graph), whose vertices are discretized values of the time series, and whose weights are the transition probabilities;
- The correlation network (whose vertices are segments $[t, t + T]$, and whose edge weights are $\text{Cor}(x_{t_1, t_1+T}, x_{t_2, t_2+T})$.

Some properties of the time series may be visible on the network (for instance, the recurrence plot of an ergodic time series represents the invariant density, in the phase space), and some graph-theoretic metrics may be informative:

- Degree (number of neighbours);
- Edge density (average degree);
- Local clustering coefficient (testing if there are more triangles around a vertex than in a random network);
- Average clustering coefficient;
- Local degree anomaly (difference between the degree of a node and the average degree of its neighbours);
- Assortativity (measuring if links tend to form between vertices of similar or dissimilar degree – this is similar to the local degree anomaly, but based on edges rather than vertices);
- Matching index (how many neighbours two vertices have in common);
- The matrix of shortest path distances; the average path length; the network diameter;
- Closeness centrality of a vertex (inverse of the average distance to other vertices);
- Betweenness centrality of a vertex (on how many shortest paths it is);
- Edge betweenness (idem, with an edge).

Financial crises and the evaporation of trust
A. Anand (2009)

Financial crises can be modeled with a *credit network*: a dynamic network, whose nodes represent lenders and whose vertices represent loans. From time to time, information about a node (bank) is revealed (leaked) and other lenders may then decide (in a game-theoretic way) to foreclose their positions with it. In this model, the only cause of default is the breakdown of trust. The opacity of the network increases the risk of a bank run: it makes it easier to fall into an “equilibrium”

leading to a bank run than to leave it or to fall into an “equilibrium” that does not lead to a bank run (there are many “equilibria”, most of which break down when information is leaked).

Crises are a financial example of **hysteresis** – trust is easy to lose, but hard to restore.

Optimal split of orders across liquidity pools: a stochastic algorithm approach
S. Laruelle et al. (2009)

Application of **stochastic approximation** (a continuous-time analogue of reinforcement learning – used here in a discrete setting) to splitting orders across several dark pools – the algorithm progressively learns which pool is the best, in terms of volume and price. The article details an alternative algorithm, based on *optimization under constraints*.

Continuous-time trading and the emergence of probability
V. Vovk (2009)

<http://www.probabilityandfinance.com/>

There are two main approaches to probability theory: the axiomatic approach (measure theory) and the frequentist approach. This article uses a third one: game theory. Even without explicitly giving a probability distribution, one can define a game-theoretic notion of “almost surely”, *i.e.*, identify events of probability 1 (on the space Ω of continuous functions $[0, \infty) \rightarrow \mathbf{R}$): “with probability one” means “unless there exists a trading strategy (it has to be adapted to the filtration \mathcal{F}_t generated by Ω – there is no measure, just a filtration) that increases the capital it risks manyfold”.

The article does not really introduce this framework, but it uses it to state and prove that the process of quadratic variation exists, and that replacing the time by this quadratic variation process gives a brownian motion – the result is known for martingales, *i.e.*, if you have a probability distribution. (Informally, this says that in *market time* log-prices are a Brownian motion.)

Nonparametric methods for volatility density estimation
B. van Es and P. Spreij (2009)

In a model of the form

$$Y = X + \text{noise},$$

if you know the distribution of Y , of the noise, and want that of X , you have a *deconvolution problem*,

$$\mathcal{D}_Y = \mathcal{D}_X * \mathcal{D}_{\text{noise}}.$$

With enough independence assumptions, kernel estimators or wavelet estimators are fine. To estimate the density of the volatility in a GARCH or stochastic volatility model, the lack of independence calls for extra caution – but kernels can still do the job.

**Admissible strategies
in semimartingale portfolio selection
S. Biagini and A. Černý (2010)**

A self-financing strategy H is a predictable stochastic process satisfying

$$H \cdot S_t = V_0 + \int_0^t H_s dS_s$$

where S is the semimartingale of discounted prices and V_0 is the initial wealth. This definition is not restrictive enough and includes undesirable or unrealistic strategies: one usually adds further restrictions, such as “never be bankrupt” or (more commonly) “ H is a supermartingale”. This article examines those restrictions and how they arise from the choice of a utility function. (The article uses the notion of Orlicz space: the L^p spaces can be generalized by replacing $|\cdot|^p$ by any non-decreasing function, for instance the lower tail of a utility function $-U(-|\cdot|)$). Strictly speaking, since the utility is not homogeneous and can even take infinite values, there are two spaces: the **Orlicz space** contains functions which have a scalar multiple of finite norm; the **Moore subspace** contains functions all of whose multiples have a finite norm.)

**Modelling financial risk
R. Pfaff
R in Finance 2010**

Overview of the tools one may need to model risk, from one or two time series.

The **PerformanceAnalytics** package provides various estimators of the most common measures of risk, value at risk (VaR) and expected shortfall (ES), e.g., assuming that the distribution is gaussian, and adding a correction for skewness and kurtosis (Cornish-Fisher).

Extreme value theory suggests to refine those estimators in one of the following three ways:

- By looking only at the tail of the data (*i.e.*, the observations above a certain threshold – the peaks-over-threshold (POT) method) and fitting it to a generalized Pareto distribution (GPD). The mean-residual-life plot (aka conditional mean exceedance (CME) plot), $E[X - a | X \geq a] \sim x$, can help select the threshold: the plot will often have a non-linear part, a linear part, and noise; you want to fit the GPD to the linear part (check the POT package and its vignette).
- An alternative, but less efficient use of the data, is to look at the distribution of the maximum return for each year or month (the block-maxima method) and fit it to a generalized extreme value (GEV) distribution (under reasonable assumptions, the limit distribution is in this family).
- Poisson Point Processes are mentioned, with no details.

You may also want to check the **VaR**, **ismev**, **QRMLib** packages (the last two accompany books on quantitative risk management).

To fit the whole distribution, you can use a generalized hyperbolic distribution (GHD, with 3 parameters in addition to the location and scale, check the **ghyp** package), or a special case, such as the normal inverse gaussian (NIG), the hyperbolic distribution (HYP, check the **HyperbolicDist** package for the pdf functions and some fitting algorithms), etc. (also check the **SkewHyperbolic** package). The **actuar** package contains common loss distributions (with fewer parameters).

Serial dependence skews the estimators or risk (VaR, ES) that rely on iid returns: check the **tseries::garch** (GARCH), **fGarch::garchFit** (ARMA-GARCH), **bayesGARCH** (MCMC estimation of a GARCH(1,1) model with Student innovations), **rgarch** (on RForge). packages. There are also some multivariate GARCH packages, as well: **ccGARCH** (conditional correlation GARCH), **gogarch** (generalized orthogonal GARCH).

To account for dependence, do not rely on correlation (there are many fallacies, such as the meaning of a zero correlation or the extreme values not being -1 and 1) but prefer the rank correlation, Kendall’s tau, the index of upper (and lower) dependence

$$\lambda_{\text{upper}}(X, Y) = P[Y > F_Y^{-1}(q) | X > F_X^{-1}(q)]$$

or a copula. (e.g., Gaussian, Gumbel, Clayton or Student, estimated by the method of moments from the rank correlation or Kendall’s tau, or by maximum likelihood). Non-symmetric copulas are not mentioned; no R packages are mentioned.

A copula-GARCH model can be estimated as follows:

- Fit the time series independently, as GARCH(1,1) models with Student innovations, S unless you have a very good reason to think that the serial dependence is negligible;
- Estimate the copula of the residuals;
- Compute the chosen risk measure (VaR, ES, draw-downs, etc.) using simulations.

**RQuantLib: interfacing QuantLib from R
D. Eddelbuettel and K. Nguyen
R in Finance 2010**

QuantLib is a C++ library, using Boost, with bindings for many languages (Python, etc. via Swig; the R binding is manual), for option pricing. implementing various products and methods (Black-Scholes, finite differences, binomial, Monte Carlo, Monte Carlo with low discrepancy sequences, etc.).

```
example(EuropeanOption)
example(BinaryOption)
demo(OptionSurfaces)
```

Fixed income products were recently added: you can build a discount curve (**DiscountCurve**, **FittedBondCurve**) and price bonds (**ZeroCouponBond**, **FixedRateBond**, etc.). A GUI (**RQuantLibGUI**), currently only for bonds, based on the **traitr** package

(inspired by Python's traits UI module) has also been added.

Extending and embedding R with C++
D. Eddelbuettel
R in Finance 2010

To speed up computations in R, you can:

- Use faster functions, that only compute the data needed and none of the diagnostics, for instance `lm.fit` instead of `lm`
- Use `.C` to call a C function: you have to cast the arguments to the right type in R, to compile the shared library (`R CMD SHLIB`) and load it (`dyn.load`); the C code only deals with pointers;
- Use `.Call` to call a C function: it is more transparent on the R side, you can use arbitrarily complicated types, but you have to cast the arguments in C from the `SEXP` type using macros, leading to hard-to-read code; you still have to compile and load the shared library;
- Use `Rcpp`, that relies on the STL to replace those macros and enhance code readability, at the expense of speed (it will be faster than pure R code, but slower than `.C`);
- Use the `Inline` package, that uses any of those three approaches, but transparently compiles and loads the shared library;
- Combine any of the above with optimized libraries, such as GSL or the C++ Armadillo linear algebra classes (the Gnu Scientific Library (GSL) is a C library, with no platform-specific optimizations and a sometimes awkward C-like syntax; it is not limited to linear algebra; Armadillo is a (LGPL) C++ wrapper around Lapack and Atlas, with (compile-time) delayed evaluation of some operations to more efficiently combine them, using templates).

RInside works in the other direction: it allows you to call R from C++.

***Statistical finance for investors
unfamiliar with quantitative methods
using stockPortfolio in R***
N. Christou and D. Diez
R in Finance 2010

Three functions to download stock returns from Yahoo; estimate the variance matrix and the expected returns; and compute "the" optimal portfolio (they mean the tangential portfolio), with or without short-selling. They provide several (mostly undocumented) estimators of the variance matrix and the expected returns:

- Sample variance matrix;
- Constant correlation model: the correlation of any two stocks is always the same;
- Multigroup model: the correlation can take two values, depending on whether the stocks are in the same industry or not; industry membership has to be provided;
- Single index model (a 1-factor model, where the risk

factor has to be provided).

***Business objectives
and complex portfolio optimization***
P. Carl et al.
R in Finance (2010)

Simple optimization problems (linear, quadratic, conical) are well addressed by the **fPortfolio** package, but for real-world problems and less trivial business objectives, such as the expected shortfall (aka conditional value at risk (CVaR), often the Cornish-Fisher CVaR computed from historical data) or the conditional drawdown at risk (mean of the worst $p\%$ drawdowns), and constraints, such as position limits (no single asset should be more than $x\%$ of the portfolio) or CVaR limits (no single asset should have a contribution to the CVaR over $x\%$), the **PerformanceAnalytics** package provides an "approximatively correct [solution] rather than [a] precisely wrong [one]", using two methods:

- Random portfolios: start with random weights and modify them so that the constraints are met (this is an optimization problem with constraints but no objective functions); this will give a set of portfolios on the boundary of the feasibility domain and some in the interior;
- Differential Evolution, *i.e.*, a genetic algorithm used to evolve a population of portfolios to optimize several objective functions (maximize return and minimize risk), so as to cover the (efficient) boundary of the feasibility domain.

Since those computations deal with many (independent) portfolios, they can easily be parallelized.

The package also provides many plots: portfolios in the risk×return space (you can compare your portfolio with an equal-weighted or *equal-risk* (equal-contribution-to-CVaR) benchmark); cumulative returns; monthly returns (and expected shortfall) over time; drawdown; portfolio weights by asset (or asset type) over time; contribution to the portfolio expected shortfall (or value at risk) by asset, over time.

RProtoBuf: Protocol Buffers for R
R. Francois and D. Eddelbuettel
R in Finance 2010

Protocol Buffers is a serialization format (like JSON, BSON (binary JSON, used by some unstructured databases, such as MongoDB), YAML, ASN, or even XML) and corresponding C++ library, developed by Google. The **RProtoBuf** packages allows you to read and write data in this format, and can be used when your data does not easily fit in a CSV file. It is possible to (manually, for the moment) use `Rcpp` to speed up data access.

***Testing, monitoring and dating
structural changes in FX regimes***
A. Zeileis

The **strucchange** package can be used to assess when a country changed the control on its currency:

- Consider FX returns with respect to a currency not involved in the affair, e.g., CHF;
- Regress the returns against those of currencies potentially involved (USD, EUR, GBP, JPY);
- Test if the coefficient for the dollar is significantly different from 1, if the other coefficients are significantly different from 0;
- Look for a regime change.

[duplicated?]

**Portfolio optimization
with CVaR budgets**

K. Boudt et al.

R in Finance 2010

The risk budget (risk is measured by the expected shortfall (ES, CVaR), estimated from historical moments via the Cornish-Fisher estimator) can be used in the portfolio construction process (with the DEOptim package):

- As an objective: minimize the CVaR of the portfolio, or minimize the maximum contribution to CVaR;
- As a constraint: equal-risk portfolio (all assets are required to have the same contribution to the portfolio CVaR, to downweigh “hot spots”) or 60/40 allocation between (the contribution to CVaR of) equities and bonds.

**Entropy correlation distance method
applied to study correlations
between the gross domestic product
of rich countries**

A. Ausloos and J. Miśkiewicz (2009)

Another distance between positive time series (for clustering, minimum spanning tree, etc.): the correlation between their moving-window entropies.

**Spiraling towards market completeness
and financial instability**

M. Marsili (2009)

There seems to be a singularity (like S. Hawking’s black holes) in the space of possible financial markets, as we increase the complexity of financial instruments to increase market completeness – this also increases any market imperfection (lack of transparency, asymmetric information, etc.).

The premium of dynamic trading

C.H.Chiu and X.Y.Zhou (2009)

The efficient frontier of a set of assets is usually computed for a single period. It can be generalized to a multiperiod (or even continuous-time) setting. The efficient frontier is (unsurprisingly) higher but is still a line if there is a risk-free asset.

**Characterizing individual
communication patterns**

R.D. Malmgren et al. (2009)

Human-generated events (e.g., sending emails) can be modeled as a 2-state Markov chain (active/inactive), the agent emitting a signal (Poisson process) with a different rate in each state.

**Personalized recommendation via integrated
diffusion on user-item-tag tripartite graphs**

Z.-K. Zhang et al.

Many recommendation algorithms are just diffusions (random walks) on bipartite (user-item and user-tag) or tripartite (user-tag-item) (hyper)graphs.

**Hypergraph topological quantities
for tagged social networks**

V. Zlatić et al. (2009)

Another article on the same subject – nothing new.

**Understanding
the Lee-Carter mortality forecasting model**

F. Girosi and G. King (2007)

The Lee-Carter mortality model approximates the matrix $m_{a,t}$ of the logarithms of mortality rates in (age) group a and year t as a rank-1 matrix (use the singular value decomposition (SVD) to find it)





$$m = \beta \gamma'$$

where β is the base mortality in each group and the mortality index γ models the time evolution of the mortality rates, often as a random walk. The resulting process $(m_{\cdot,t})_t$ is a multi-dimensional random walk with strong constraints on the variance matrix. If the rank-1 assumption is not valid (look at the eigenvalues), you may want to consider a rank- k approximation or an unconstrained random walk.

**The Lee-Carter model
for forecasting mortality, revisited**

S.H. Li and W.S. Chan

The Lee-Carter model is not robust. You may want to add a contamination model that looks for:

- Additive outliers: 
- Innovational outliers: 
- Level shifts: 
- Temporary changes: 

**The role of a matchmaker
in buyer-vendor interactions**

L. Lü et al. (2009)

Around the stable marriage problem: match men/women, employers/job seekers, landlords/tenants, buyers/sellers to maximize some total utility

$$u = (x^y + y^k)^{1/k}$$

(changing k creates more or less inequality). The “matchmaker” is just a personification of the perfect knowledge of all utilities, as opposed to auctions or the (potentially never-ending) quest for a sufficiently good match.

***A dynamic model of
time-dependent complex networks***
S.A. Hill and D. Braha (2009)

We have acceptable models of (random, static) networks, but few of *dynamic* networks. Here is one, in which the hubs (highly connected nodes) can change quickly, as observed in daily email datasets:

- Start with a (static) underlying network;
- For each day, build a subnetwork via a node-reinforced random walk: choose the next node of the random walk among the neighbours of the current node, with probabilities

$$p \propto 1 + \lambda \times \text{number of visits.}$$

***Effective and efficient similarity index
for link prediction of complex networks***
L. Lü et al. (2009)

One can try to predict missing or future links in a network (the recommendation system of an online shop is a link prediction system in the bipartite client \times item graph; there are applications in gene \times protein networks as well) by looking at node similarity measures, such as:

- The number of common neighbours $(A^2)_{xy}$;
- The *Katz index*, *i.e.*, the weighted number of paths between two nodes, with less weight on longer paths,

$$\left(\sum_{k \geq 2} \beta^k A^k \right)_{xy};$$

- The first two terms of this series, $(A^2 + \varepsilon A^3)_{xy}$.

***Limits of declustering methods for
disentangling exogenous from endogenous
events in time series with foreshocks, main
shocks and aftershocks***
D. Sornette and S. Utkin (2009)

A time series of events (earthquakes, financial crises, etc.) can be modeled as a *Hawkes process* (aka ETAS (epidemic-type aftershock) model)

$$\lambda(t, M, \mathcal{H}_t) = \mu(t, M) + \sum_{t_i < t} h(t, M, t_i, M_i)$$

where M is the magnitude of the event, the first term is the background density (alone, it would lead to a Poisson process), the second term is the response function to past events (measuring their propensity to trigger aftershocks). Algorithms to disentangle those events (*i.e.*, label them as “exogenous” (background) or “endogenous” (aftershock)) do not perform well.

***Detecting network communities
by propagating labels under constraints***
M.J. Barber and J.W. Clark (2009)

The label propagation algorithm (LPA) to identify communities (assign a label to each vertex at random; change the label of a vertex to that of the majority of its neighbours, iterate until convergence) finds a local optimum of some optimization problem – it turns out that the global solution is the situation in which all vertices have the same label... A penalty can be added to bring the optimization closer to what we expect.

Structure of shells in complex networks
J. Shao et al. (2009)

Exercise on generating functions and networks; the shell (wrt a node) is the set of nodes ℓ edges away from it.

Market bubbles and crashes
T. Kaizoji and D. Sornette

A readable, non technical, superficial review article:

- Bubbles are frequent and follow the following pattern: new opportunity (beware when the media start to talk about “new ...” – you could try to build a bubble index by counting the number of occurrences of the word “new” in the press – add synonyms, as well), rising prices, investments in increasingly illiquid assets, crash.
- Markets are not efficient: current prices are not the present value of future cash flows.
- Evidence that this can be explained by limits on arbitrage (e.g., short sales restrictions) are mixed: some look convincing (such restrictions should lead to overvaluation), but so is the evidence that those limits can be circumvented (by using options instead of short sales: the put-call parity, from tick data (rather than close prices, which are quotes on which one cannot trade), is valid);
- Rational investors do ride bubbles; by anticipating them, they actually create them: they know that noise (irrational) traders will follow them;
- Rational investors do not arbitrage bubbles out, because then do not know *when* to do so – the bubbles therefore continues to grow;
- Physicists use statistical mechanics to model markets as the aggregation of many “agents” and describe bubble bursts as phase changes;
- Bubbles are characterized by faster-than-exponential growth

$$\log(\text{price}) = \alpha + \beta(t_0 - t)^\gamma + \text{noise}, \quad \gamma < 1, \quad \beta < 0$$

(this is often visible on a 6-month window; you could use this to build a bubble detector)

- ODE with “explosive” behaviour can be generalized to SDEs (“stochastic finite time singularities”);
- The log-periodic power law model is just mentioned.

Financial bubbles, real estate bubbles, derivative bubbles and the financial and economic crisis

D. Sornette and R. Woodard (2009)

Another similar review article, with historical data,

- To explain the causes of the crises, do not spend your time identifying the first domino to fall: instead, try to understand why the dominoes were laid out like that.
- Technical and rational mechanisms (what is rational for a single investor need not be so for the market as a whole) play a small role, behavioral mechanisms dominate.
- More details on the log-periodic power law, and actual examples

$$\log(\text{price}) = \alpha + \beta(t_0 - t)^\gamma (1 + \varepsilon \cos(\omega \log(t_0 - t) + \phi))$$

Complex systems: from nuclear physics to financial markets

J. Speth et al. (2009)

Log-periodic oscillations (à la Sornette) can result from a **Weierstrass random walk**: take a step of length $b^j a$ with probability proportional to $1/M^j$ ($b > 1$, $M > 1$, vary $j \in \mathbb{N}^\times$).

The Markov modulated Poisson process and Markov Porsson cascade with applications to web traffic modelling

S.L. Scott and P. Smyth (Bayesian Statistics, 2003)

A Markov modulated Poisson process (MMPP, or Markov Poisson cascade, when the states are ordered to make the model identifiable) is a Poisson process whose rate varies according to a (finite-state, hidden) Markov process, used to model events with irregular bursts of activity (an n -state MMPP is equivalent to a superposition (sum) of n (fixed-rate) Poisson processes, some of which may be inactive, the inactivity being controlled by an n -state Markov chain). They can be fitted with the EM algorithm or MCMC data augmentation.

Causal links between US economic sectors
G.H.T. Lee et al. (2009)

To analyze a bubble or crash (after the fact), take hourly sector log-returns, segment each of them by recursive regime change tests (likelihood ratio test of a gaussian distributions versus two gaussian distributions; you can reoptimize the segment boundaries at each step), stop when some threshold is reached, assign the segments to either a “high” or a “low” volatility regime, plot all segmented time series (the segments are different for each sector) with a level of grey indicating the volatility level, try to spot the beginning of the crisis and the recovery (a month of high, resp low, volatility), compare the start and end of the crisis for

each sector – often, the first to enter is also the last to leave.

The Stress VaR: a new risk concept for superior fund allocation

C. Coste et al. (2009)

The risk of a (portfolio of) hedge fund(s) can be estimated as follows:

- Select several good 1-factor risk models (rather than a single multi-factor model – this can be seen as a form a bayesian model averaging (BMA) or a regularized multifactor model);
- Take the *maximum* of the value-at-risk (VaR) predicted by those models (rather than the VaR of their linear combination).

Adaptive model for recommendation of news

M. Medo et al. (2009)

Most recommendation systems use a *global* rating system, and only narrow down the centers of interest of the users by categories or keywords – some distinguish between short- and long-term interests; some use implicit ratings (based on access or reading time). The authors use an epidemic-like process to propagate relevant news among users with a similar interest profile; the timeliness of the news is ensured by the exponential spreading of epidemics (if the users find the news relevant) and a continuous time decay.

Financial bubble analysis with a cross-sectional estimator

F. Abergel et al. (2009)

The authors claim that the proportion of stocks with performance beyond some threshold z since “some” reference date (and its asymptotic behaviour $\sim_{z \rightarrow \infty} z^{-\alpha}$) can be used as a bubble indicator – but then, they only check that the cross-sectional variance can be used as a bubble indicator.

Stability analysis with applications of a two-dimensional dynamical system arising from a stochastic model for an asset market

V. Belitsky and A.L. Pereira (2009)

To examine the stability of stock markets, you can build a 2-dimensional dynamical system for the price and the excess demand (for a single asset).

Joint modelling of gas and electricity spot prices

N. Frikha and V. Lemaire (2009)

Gas and electricity prices are important for power plants and energy companies: they represent their costs and revenues. They exhibit the following features: seasonality, trend, spikes, heavy tails, autocorrelation, long-memory, cross-correlation. (After removing the trend and seasonality), they are often mod-

eled by Ornstein processes

$$X(t) = \sum_i Y_i(t)$$

$$dY_i = -\lambda_i Y_i dt + dL_i$$

where the L_i are brownian motions (leading to Ornstein-Uhlenbeck processes) or Lévy processes (to model the spikes – one could also use OU processes with very quick mean reversion); some of those L_i appear in both the gas and electricity models, to account for cross-correlation. The authors replace the Lévy processes by considering OU processes and more general diffusions

$$dY = -\lambda Y dt + \sigma(Y) dW$$

where σ is chosen to give the observed stationary distribution (if you know the drift and the stationary distribution of a diffusion, you can compute its volatility), modeled as a (quasi-saddlepoint approximation of a) normal inverse gaussian (NIG) distribution.

Eroding market stability by proliferation of financial instruments
F. Caccioli et al. (2009)

The proliferation of financial instruments make the markets more efficient, more complete, but it also brings them closer to an unstable state – a phase transition. This comes from the interplay of supply and demand and the price impact of transactions.

The article also contains a concise definition of financial markets (markets that allow intertemporal exchanges of wealth) and a clear description of the 1-period asset pricing framework:

- There are two times, $t = 0$ (today) and $t = 1$ (tomorrow);
- There is a finite set Ω of possible states ω at time $t = 1$, each probabilities p_ω ;
- There are K risky assets, that cost 1 today and pay $r_{k,\omega}$ tomorrow;
- There is one risk-less asset, that costs 1 today and pays 1 tomorrow;
- There is no arbitrage, *i.e.*, there is no portfolio whose return is positive or 0 in all states and positive in at least one state;
- This implies the existence of an *equivalent martingale measure* q such that

$$\forall k \quad E_q[r_k] = \sum_{\omega} q_{\omega} r_{k,\omega} = 0$$

- A contingent claim is a contract that pays f_{ω} in state ω ; if there exists a replicating portfolio θ , *i.e.*, a portfolio θ such that

$$\forall \omega \in \Omega \quad f_{\omega} = f_0 + \sum_k \theta_k r_{k,\omega}$$

the contingent claim is said to be *marketable* and its price is $V_F = E_q[f] = \sum_{\omega} q_{\omega} f_{\omega}$;

- The market is complete if there are at least $\#\Omega$ independent vectors among r_1, \dots, r_K ; then, all claims are marketable and the risk-neutral measure is unique.

A general “bang-bang” principle for predicting the maximum of a random walk
P.C. Allart (2009)

The best predictor (adapted stopping time) of the maximum (on $[0, T]$) of a random walk with drift is obtained with $\tau = 0$ if the drift is negative and $\tau = T$ if it is positive.

Risk concentration and diversification: second-order properties
M. Degen et al. (2009)

First order (I would say zeroth order) approximations of the **risk concentration**

$$C(\alpha) = \frac{\text{VaR}_{\alpha} \sum_i X_i}{\sum_i \text{VaR}_{\alpha} X_i}$$

or the risk diversification $1 - C(\alpha)$, for instance,

$$\frac{F(tx)}{F(t)} \xrightarrow{t \rightarrow \infty} x^{-1/\xi} \implies C(\alpha) \xrightarrow{\alpha \rightarrow 1} n^{\xi-1}$$

are not sufficient: the convergence is too slow. One can devise a second-order (first-order) approximation.

In the case of non-coherent risk measures (such as the value-at-risk), the diversification benefit can even be negative.

```
N <- 1e6
library(actuar)
x <- rburr(N,1,1)
y <- rburr(N,1,1)
f <- function(z) quantile(z,.99)
f(x+y)/(f(x)+f(y)) # > 1...
```

Adaptive model for recommendation of news
M. Medo et al. (2009)

Most recommendation systems use a *global* rating system, and only narrow the centers of interest of the user by categories or keywords – some also distinguish between short- and long-term interests; some use implicit ratings (based on reading time or access). The authors use an epidemics-like process to propagate relevant news among users with a similar rating profile; the timeliness of the news is ensured by the exponential spreading of epidemics (if the users find it relevant) and a continuous time-decay.

A coupled markov chain approach to risk analysis of credit default swap index products
R. Hochreiter and D. Wozabal (2009)

Confusing attempt to add some dependency (sector membership) to the transition matrix generating rating changes:

- Each assets's rating is a Markov chain (with states AAA, AA, etc.) whose transition matrix is known;
- If two assets are in the same sector, their rating are not independent and can be modeled by a Bernoulli mixing model.

***Universal power laws
in the threshold network model:
a theoretical analysis
based on extreme value theory***
A. Fujihara et al. (2009)

Asymptotic analysis of the **threshold network model** – random networks on a set of nodes obtained by randomly assigning a weight to each node (according to a given probability distribution) and linking any two nodes when the sum of their weights exceeds a given threshold.

***Leverage causes fat tails
and clustered volatility***
S. Thurner et al. (2010)

Limits on leverage cause funds to sell in falling markets, amplifying the fall. Limits based on volatility (allow a high leverage when the volatility is low) are even worse. (The article also recall what a *margin call* is: what you have to pay to a lender when the value of your collateral drops below an agreed limit.)

***Atmospheric complexity
or scale by scale simplicity?***
**S. Lovejoy et al.
Geophysical Research Letters (2009)**

The fractal nature of the atmosphere (similar structures are present at different scales) could simplify weather forecast – the article only presents evidence of these fractal structures, but no applications. [This structure can be seen as a form of symmetry, \mathbf{Z} acting by $n : \mathbf{x} \mapsto \lambda^n \mathbf{x}$, which should lead to a conservation law (Noether's theorem), that could simplify the PDEs – or simply allow macroscopic forecasts without looking at the microscopic data.]

***Correlation breakdown,
copula credit default models and arbitrage***
R. Tzani and A.P. Polychronakos (2009)

Complete nonsense: when the market and the model (Gaussian copula for CDSes...) disagree, the market is wrong and the model is right – and you are bankrupt.

A computational view of market efficiency
**J. Hasanhodzic et al.
AlphaSimplex, MIT (2009)**

In a market with periodic returns, modified by the impact of bounded-memory traders, traders with more memory make more profits.

***Market impact and trading profile
of large trading orders in stock markets***
E. Moro et al. (2009)

Market impact of hidden orders (large trading orders, executed incrementally) increases as the square root of the order size; once the order is finished, the impact reverts to half its peak value

The scale of market quakes
T. Bisig et al. (2009)

The multiscale structure of log-price time series can be described by the “term structure of volatility” (volatility of the time series of returns over intervals of size τ , as a function of τ); it is a function, but its values can be combined in some ad hoc way to produce a single numeric “scale of market shocks”. The authors present (somewhat confusingly) a similar measure based on the irregular time series of “directional changes of amplitude greater than λ ”, for various values of λ . This is similar to the term structure of volatility, but measured on *market time*, rather than clock time.

***Modeling scientific citation patterns
and other triangle-rich acyclic networks***
Z.X. Wu and P. Holme (2009)

Empty article presenting a model of network evolution, whose parameters are: the out-degree distribution; the aging of the relevance of papers (nodes); the formation of triangles.

***On the relationship between
trading network and WWW network:
a preferential attachment perspective***
A. Mirzal (2009)

The **page rank** algorithm (the rank of a page depends on the rank of the pages pointing to it, with less weight for pages with too many links),

$$\text{page rank}_i = \sum_{j \rightarrow i} \frac{\text{page rank}_j}{\text{out-degree}_j}$$

is an eigenvalue problem

$$p = p \cdot O^{-1}L$$

p : vector of ranks

O : diagonal matrix of out-degrees

L : incidence matrix

often solved iteratively (after some rescaling to ensure convergence).

The HITS (hypertext-induces topic search) algorithm considers two types of ranks, “hub” and “authority” (you can see this as a hidden fuzzy bipartite structure)

$$\begin{aligned} \text{authority}_i &= \sum_{j \rightarrow i} \text{hub}_j \\ \text{hub}_j &= \sum_{j \rightarrow i} \text{authority}_i \end{aligned}$$

is another eigenvalue problem

$$\begin{aligned}\text{authority} &= \text{hub} \cdot L \\ \text{hub} &= \text{authority} \cdot L\end{aligned}$$

i.e., the vectors of authority and hub scores are eigenvectors of $L'L$ and LL' (to ensure unicity and convergence, it may be necessary to rescale and/or regularize those matrices, as with ridge regression).

The same ideas can be used to study *trading networks*, *i.e.*, networks of buyers, sellers and resellers. [There may be a link between the price discovery process and the convergence of those algorithms.]

A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis

D.W. Witten et al.

The first r components of the singular value decomposition of a matrix X ,

$$X = UDV', \quad U'U = 1, \quad V'V = 1, \quad D \geq 0,$$

give its best rank- r approximation (for the Frobenius norm)

$$\sum_{k=1}^r d_k \mathbf{u}_k \mathbf{v}_k' = \underset{Y, \text{rank } Y \leq r}{\text{Argmin}} \|X - Y\|_F^2.$$

This can be generalized by adding a penalty for \mathbf{u} and \mathbf{v} , leading to a penalized rank- r approximation, and a **penalized matrix decomposition** (PMD). The suggested penalty combines the lasso (an L^1 penalty, $\sum_u |u_i|$) with the *fused lasso* ($\sum |u_i - u_{i-1}|$).

The PMD can be used to define **sparse principal components** (other formulations exist, based on regression properties or the variance minimization property of the principal components).

The article also provides (efficient) algorithms to compute those decompositions.

Sparse inverse covariance estimation with the lasso

J. Friedman et al. (2007)

A sparse variance matrix estimator, based on the lasso (L^1 penalty); the sparsity of the matrix gives an undirected (sparse) graph between the variables.

Ant-based clustering and topographic mapping

J. Handl et al.

Ant-based sorting is a topographic mapping algorithm (*i.e.*, an algorithm that maps observations from a high-dimensional space to a plane or some other low-dimensional space, such as multi-dimensional scaling (MDS) or self-organizing maps (SOM)) in which agents (ants) pick up observations (randomly), transport and drop them (randomly) where there are similar observations around. The algorithm can be improved as follows:

- Increase the penalty for dissimilarity, to help separate the clusters;
- Add some short-term memory, so that each ant knows if a newly picked up observation is similar to those around a previous drop point;
- Progressively increase the radius of perception of the ants;
- For a limited time, take into account the number of occupied cells (the density), to help spread out the clusters (only for a limited time, because we want the clusters to be globular, cluster-like);
- The article also suggests some empirical changes to the pickup and drop probabilities, and gives some advice on the choice of the parameters);
- If you want clustering, and not only topographic mapping, apply some clustering algorithm such as agglomerative clustering to the result.

The performance of the algorithm was examined with the F measure (from the ideas of *precision* and *recall* in *information retrieval*)

$$\begin{aligned}G_i &= \text{desired groups} \\ C_i &= \text{clusters produced} \\ p_{ij} &= \frac{\#G_i \cap C_j}{\#C_j} \\ r_{ij} &= \frac{\#G_i \cap C_j}{\#G_i} \\ F_{ij} &= \frac{2p_{ij}r_{ij}}{p_{ij} + r_{ij}} \\ F &= \sum_i \frac{\#G_i}{\#X} \text{Max}_j F_{ij},\end{aligned}$$

the Dunn index

$$D = \text{Min} \frac{\text{inter-cluster distance}}{\text{Max cluster diameter}}$$

and the intra-cluster variance.

For clustering, the algorithm fails to recognize the clusters in presence of a hierarchical structure (it only recognizes the top of the hierarchy), fails to recognize small clusters in general; but scales much better than competing algorithms. For topographic mapping, MDS performs better in low dimensions, SOM in high dimensions.

A hybrid particle swarm ant colony optimization for design of truss structures

A. Kaveh et al. (2008)

(This is just a random article on the subject, among hundreds of others: it does not bring anything new.) **Particle swarm optimization** (PSO) is a population-based optimization algorithm, in which the particles (candidate solutions) move, influenced by

- Their momentum;
- The best solution they have found so far;
- The position and momentum of the particles around them, *i.e.*, (indirectly) the best solution their neighbours have found so far;

- In the **ant colony optimization** (ACO) variant, they are also influenced by (pheromones left around, indicating the momentum of) particles that were there some time ago.

This may not work well in high dimensions, because of the curse of dimensionality, except if, as here, *constraints* drastically reduce the search space.

Choosing colors for Data Visualization **M. Stone (2006)**

- We perceive contrasts of colours in three dimensions: hue, value and chroma; try to use all three.
- Excessive hue variations create clutter: also use contrasts of value and chroma.
- Avoid saturated colours (only use them to highlight).
- To be legible, graphical (or textual) elements should have a contrast of *value* with the background (contrast of hue or chroma is insufficient); smaller sizes require more contrast. Indeed, some artists first “block in the values” and only add colour at a later stage.
- Only use 2 or 3 hues, e.g., analogous or complementary or split-complementary (*i.e.*, one colour and two colours close to its complementary).
- A white background is preferable: it provides a reference for colours (the “white balance” of your digital camera).

Dot Plots: A Useful Alternative to Bar Charts **N.B. Robbins (2006)**

Barcharts can become cluttered when there are many observations (the ink/information ratio is high), when there are several variables (stacked bar charts are almost always a bad idea), or when the zero is not meaningful or too far from the bulk of the data: in those cases, you can use dotplots instead.

Practical rules for using colors in charts **S. Few (2008)**

A list of simple rules, with examples and counterexamples:

- Use a consistent background, to allow a reliable comparison of the colour or value of the plot elements (think of grey squares of the same intensity, on a grey *gradient*);
- The colour of the objects should contrast with the background (counterexample: a heatmap, *i.e.*, a table whose cells are coloured, where the important values sometimes end up in black on a dark background);
- Only use colour with a purpose in mind;
- The differences in colour should be meaningful (the viewer will instinctively try to interpret those differences: at the very least, this would distract him from what he should be looking at; the typical counterexample is a barchart with a different colour for each bar);

- Use neutral colours (greys, browns, pastels) for most information, and bright/dark saturated colours for what requires more attention;
- Do not cover large areas with saturated colours: that looks garish, unsophisticated, unprofessional (the Brewer palettes are fine for large areas);
- Small elements (thin lines, dots in a scatterplot) require brighter/darker and more saturated colours to be easily distinguished; this is the reason why the Brewer palettes (designed for maps) are inadequate for line plots or scatterplots;
- Use a single hue to encode sequential values (or two for diversifying ones, *i.e.*, if there is a clearly defined “zero”);
- Use a dull, hardly visible colour for less important elements (axes and borders could be grey, the background could be white);
- Do not use red and green together, even if they seem widely understood as “traffic light colours”: most colourblind people cannot distinguish them (red and blue is fine);
- Choose a few palettes in advance (and for each, a variant with more saturated colours for scatterplots and lineplots) and keep them at hand;
- Do not use visual effects (3D, reflections, etc.).

Introduction to cycle plots **N.B. Robbins (2008)**

For progressively changing periodic patterns:

```
xyplot( x ~ week | day_of_week )
```

Sequential implementation of Monte Carlo tests with uniformly bounded resampling risk **A. Gandy**

The power of a Monte Carlo test can be improved by replacing the sum

$$\sum_1^n \mathbf{1}_{T_i \geq T}$$

with the whole path of the partial sums

$$\sum_1^k \mathbf{1}_{T_i \geq T}, \quad k \geq 1,$$

and comparing them with boundary paths.

A simple forward selection procedure based on the false discovery rate control **Y. Benjamini and Y. Gavrilov (2009)**

Variable selection, in a regression model, can be seen as a penalized regression with an L^0 penalty; this can be generalized to other penalties, including non-linear ones (a non-linear function of the L^0 or L^1 norm of the coefficients). Variable selection algorithms often perform statistical tests to decide whether to add a new

variable, all with the same threshold for the p -value; instead, one can use the false discovery rate,

$$\text{FDR} = \frac{\#\text{true rejected null hypotheses}}{\text{rejected null hypotheses}}$$

to progressively change this threshold.

***Bayesian methods
for measuring operating risk***
C. Alexander (2002)

If you do not have enough loss data to fit an extreme distribution, add a bayesian prior, in the form of a bayesian network (*i.e.*, more variables, with known dependency relations between them), to incorporate domain knowledge.

***Retrospective exact simulation
of diffusion sample paths with applications***
A. Beskos et al. (2006)

Most algorithms sampling from the paths of a diffusion do not sample from the stochastic process itself, but from a different but hopefully close discrete process. The (confusing) article presents an exact simulation algorithm, based on rejection sampling.

Exact simulation of diffusions
A. Beskos and G.O. Roberts (2006)

Still not understandable, in spite of the examples.

***MCMC methods for multi-response
generalized linear mixed models:
the MCMCglmm R package***
J. Hadfield
Journal of Statistical Software (2006)

The MCMCglmm package fits generalized linear mixed models; it is more restricted than Bugs/Jags, but faster. Presenting the same example in the Bugs language (which is straightforward to understand) would help understand the syntax of the function.

***Exploratory data analysis leading towards
the most interesting binary association rules***
A. Iodice D'Enza et al.

Given a binary dataset, *e.g.*, a basket \times item boolean matrix Z representing the sales of an online shop, **association rules** are pairs of items, $A \implies B$, with a large enough **support** $P(A \cap B)$ and a large enough **confidence** $P(B|A)$ – this can be generalized to tuples. Association rule mining algorithms, such as the *Apriori* algorithm, usually select all pairs with a large enough support, and among them, take those with the largest confidence. The computations can be formulated in linear algebraic terms,

$$\text{Support} = \frac{1}{n} Z'Z$$

$$\text{Confidence} = Z'Z \text{diag}(\text{Support})^{-1}$$

but the matrices are too large. Dimension reduction through **incremental k -means** (add the baskets one by one) can make (those matrices block-diagonal and) the computations amenable.

***Volatility forecasts and the at-the-money
implied volatility: a multi-components ARCH
approach and its relation with market models***
G. Zumbach
Risk Metrics (2007)

A **multiscale ARCH** process is an ARCH(∞) process of the form

$$r_{t+1} = \sigma_t \varepsilon_{t+1}$$

$$\varepsilon_t \sim N(0, 1)$$

$$\sigma_t^2 = \sum_{k=1}^n w_k \sigma_{t, \tau_k}^2$$

$$\sigma_{t, \tau}^2 = e^{-1/\tau} \sigma_{t-1, \tau}^2 + (1 - e^{-1/\tau}) r_t^2$$

i.e., the squared volatility is some linear combination of exponential moving averages, with different time scales, of the squared returns. These factors play a role similar to the risk factors of a risk model. One or two different time scales (say, 3 weeks and 3 years) may suffice. The long-memory ARCH uses a large number of components, but the characteristic times and the weights have simple forms (geometric increasing and logarithmic decreasing),

$$\tau_t / \tau_{k-1} = \rho$$

$$w_k \propto 1 - \frac{\log \tau_k}{\log \tau_0}$$

This can be generalized to a continuous-time model.

***Stability of graph communities
across time scales***
J.-C. Delvenne et al. (2009)

Gauging the quality of a clustering (community structure) in a graph is difficult: here is another measure to consider. Let $(X_t)_t$ be the random walk on the graph and $T_{i,t} = \mathbf{1}_{X_t \in C}$ the boolean variable denoting membership to community C (it is no longer Markov). Its autocorrelation function (plot it with a log-scale for the lag) is a measure of the stability of the community.

Mapping change in large networks
M. Rosvall and C.T. Bergstrom (2009)

You can assess the significance of a clustering or partition of a network (a kind of dimension reduction) by comparing with that of bootstrap samples; the evolution of those clusters can be represented in an **alluvial diagram**. This is exemplified by the emergence of neuroscience from molecular biology, psychology, psychiatry, neurology, medicine.

Random hypergraphs and their applications

G. Ghoshal et al. (2009)

Folksonomies (sets of $\langle \text{user, resource, tag} \rangle$ triplets, in collaborative tagging websites such as flickr, citeU-like, delicious, bibsonomy) or RDF databases (sets of $\langle \text{subject, object, verb} \rangle$ triplets) can be represented by **tripartite hypergraphs**. Some care is needed to extend classical graph-theoretic notions: the degree of a vertex is the number of hyperedges containing it (rather than the number of neighbours of a given colour); there is a degree distribution for each vertex type (colour) and the three are linked (they cannot be chosen independently: they must imply the same number of hyperedges); there are various kinds of projections (onto one or two colours, using vertices of one or two colours). Some properties of hypergraphs can be studied by looking at the generating functions of the three degree distributions: degree distribution of the projections, existence of a giant component, percolation (*i.e.*, proportion of vertices you can remove without breaking the giant component). A random tripartite graph can be defined by specifying the degree distribution (they have to be compatible, *i.e.*, they should all suggest the same number of hyperedges), choosing hyperedge stubs at random for each vertex, and joining them at random. The proportions of those random tripartite hypergraphs do not match those of real-world ones, mainly because they fail to account for multiple tagging (the same user giving different tags to the same resource).

The empirical properties of large covariance matrices

G. Zumbach (2009)

When estimating a covariance matrix from time series, you may want to estimate it on a moving window (or with decaying weights) and consider the evolution of:

- The top eigenvalues (spectrum);
- The distribution of the eigenvalues (spectral density);
- The subspace spanned by the top eigenvalues (it can change, even if the eigenvalues look stable);

In the spectrum, there is no clear separation between noise and non-noise eigenvalues as suggested by random matrix theory (RMT). The dynamics of the covariance and the correlation matrix are very different (some people suggest they have significantly different characteristic time scales and suggest to model correlation and volatility separately, as in the constant correlation GARCH model, with a quickly moving volatility and a slowly changing correlation).

Executing large orders in a microscopic market model

A. Weiss (2009)

Comparison of a microscopic market model (that explicitly models the limit order book, or even a generalized order book, which contains the opinion of a

fair price for all investors, to account for hidden liquidity) and a macroscopic model (that models the limit order book more implicitly, *e.g.*, with two parameters, “shape” and “resilience”): how do **optimal trading strategies** for one model fare in the other? This is an (imperfect) way of measuring the robustness (to model specification) of those strategies.

Computational modeling of collective human behavior: example of financial markets

A. Kirou et al. (2008)

A market model in which traders use information both global and local (from their social network).

Studies of the limit order book around large price changes

B. Toth et al. (2009)

After a large price jump (a single-stock event, not a market crash), all quantities progressively go back to normal, following a power law. This relaxation can be simulated by a multiagent model.

Production copula

H. Iyetomi et al. (2009)

The *production equation*

$$\text{Production} = F(\text{Labour}, \text{Capital})$$

(*e.g.*, the Cobb–Douglas model, $\text{Production} \propto \text{Labour}^\alpha \text{Capital}^\beta$) should be replaced by (a statistical model or, more generally) the probability distribution of

$$(\text{Production}, \text{Labour}, \text{Capital})$$

to account for the heterogeneity of economic agents. The article models the marginal distributions and the copula separately – but their model has fatter tails than the data... Their copula is a **non-exchangeable** (*i.e.*, hierarchical) Gumbel copula.

Smoothing of multivariate data Density estimation and visualization

J. Klemelä

Wiley (2009)

In this multivariate data analysis book, the author uses two types of plots to display information, in very different contexts:

- **Rooted trees**, whose nodes are labeled by numbers, such as the heights of the nodes (for nice plots it should increase as you move away from the root) or the n th coordinate of the point (if it came from the dataset);
- **Volume plots**, built from a rooted tree whose nodes are labeled with two numbers interpreted as the altitude of the node and its width; the tree plot looks like stacked slabs of tree. (It is not uniquely defined: each time there is a branching, you can choose the order of the branches.) This is a 1-dimensional analogue of a treemap;

Here are a few examples.

Plotting the barycenter of each connected component of each level set $[f = \lambda]$ or upper level set $[f \geq \lambda]$ of a density f versus λ gives a tree. Plotting a segment for each connected component of each level set $[f = \lambda]$ (of length the volumes of the level sets, suitably renormalized, e.g., by using the radius of a ball with the same volume), and stacking those segments, gives a **volume plot**: it displays the modes and spreads of the original distribution (the notion of “mode-preserving transformation” or *mode isomorphism* can be formalized). The shape of the peaks gives information about the skewness and the fatness of the tails. The volume plot is actually a 1-dimensional analogue of the treemap.

In a volume plot, you can replace the volume of the levelsets by any interesting function. You can also replace the density by any function built from the data at hand – think of all the functions you use in scagnostics plots. For instance, if you have a shape (say, a single level set), the distance from a reference point gives a foliation, called the *radius transform* of this shape. The corresponding tree, the **shape tree**, formed from the connected components of $S \setminus B(P, r)$ when r varies, shows the **appendages** of the shape. You can change the reference point to hide or reveal more appendages; you can also change the metric. This is called a **tail plot** (or a **radius plot**).

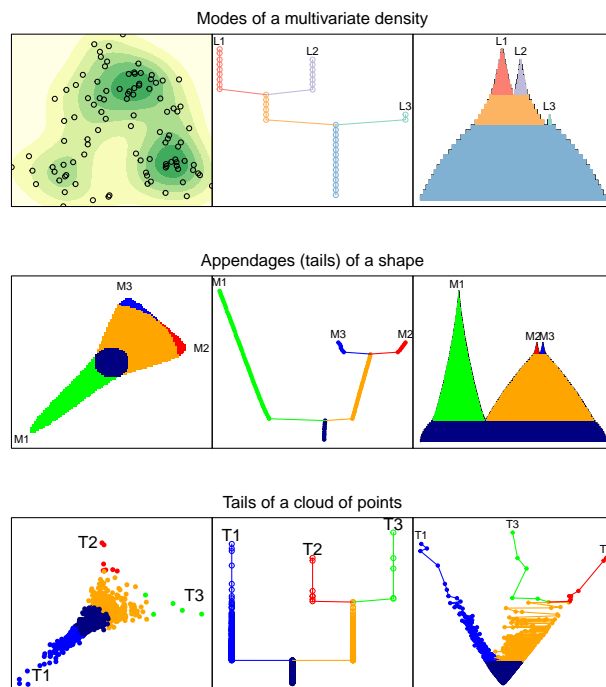
With a cloud of points, you can remove and plot the most central point, and iterate, until the cloud ceases to be δ -connected (a set is δ -connected iif for any two points, there is a chain of points going from one to the other, so that the distance between two consecutive points be less than δ – the problem is that you have to choose δ – instead, you could cluster the remaining observations); you then process the components separately. When plotting this **tail tree** (e.g., using the distance from the center versus the i th coordinate of the last point removed), it is more insightful to use the observations removed rather than the barycenters of the ρ -connected components. This can help guess the number of tails and their shapes: the book provides a large list of examples, in dimension 2, for common copulas. The number of tails concerns me: in dimension d , 2^d tails is not uncommon. Since a tail tree is just a tree structure on a set of points, one could consider other tree structures, such as the minimum spanning tree (MST) or the shortest-path tree (a rooted analogue of the MST).

With a density (or its copula), the tail plots of the level sets can be arranged into a perspective plot.

With a cloud of points, you can estimate the density and look at the mode plot. However, density estimates usually depend on a smoothing parameter: by letting it vary, you have a family of volume plots, that can be represented as a perspective plot.

To visualize the effect of the smoothing parameter, you can also take the modes of the density estimate for different values of the smoothing parameters and arrange

them into a tree



The R code is available in the **denpro** (for the plots, which are very tricky to fine-tune – no ... argument, half-missing contours) and **delt** (for the density estimation algorithms) packages.

The first part of the book also reviews many types of statistical plots and descriptive statistics for high-dimensional datasets. The **depth** of a point (in a cloud of points, or wrt a density) can be defined as $(1 + d)^{-1}$, where d is the Mahalanobis or Euclidian L^2 or L^1 distance to the “center”; as the half-space depth (the minimum proportion of points in a half space containing your point); as the simplicial depth (the probability that your point is inside a simplex whose vertices are random points from the cloud or the distribution); as the convex hull peeling depth (points on the convex hull have depth 1; remove them; points on the convex hull of the remaining points have depth 2; etc.).

Functions or probability densities can be examined by pp-plots, qq-plots, ff-plots; perspective or contour plots (but these are not adequate to visualize tails); slices, projections (marginal densities), or more generally Radon transforms (tomography).

The spread can be examined by looking at the volume of the upper level sets

$$\lambda \sim \text{Volume}(\{x : f(x) \leq \lambda\}).$$

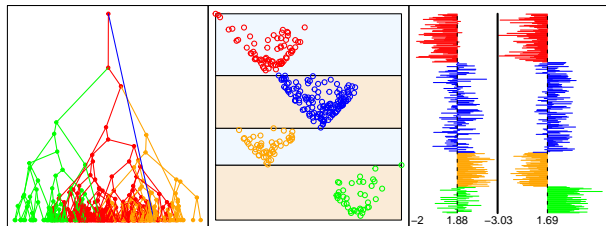
You may want to normalize those volumes (replace them by the radius of a ball of the same volume); you may want to replace the sequence of nested sets $f \leq \lambda$ (the inspiration comes from **Morse theory**) by $[f = \lambda]$ or by depth regions or by minimum volume sets (sets of minimum volume with a given probability).

Multivariate quantiles are defined by replacing $p(X - u)^+ + (1 - p)(X - u)^-$ in the definition of a

quantile by

$$\|X - u\| + p\langle v, X - u \rangle, \quad p \in]0, 1[, v \in \mathbf{S}^d.$$

The book suggests a few plots to examine clusters, but they were not convincing. For instance, you can plot the depth of a point against its i th coordinate, and add edges. In a parallel level plot, you choose the number of clusters and plot the clusters separately in this way, on top of each other – it will always look informative, even if your choice of cluster number is wrong. **Graphical matrices** present the same problem.



The second part of the book is very technical and explains how to prove inequalities and establish convergence results for density estimators.

Among the interesting technicalities are the correlation between a real-value and a vector valued random variable,

$$\text{Cor}(X, \mathbf{Y}) = \max_{\phi} \text{Cor}(Y, \phi' \mathbf{X}),$$

the **mixing coefficient**, $\beta(X, Y) = \frac{1}{2} \int |f_{X,Y} - f_X f_Y|$, can also be computed as

$$\frac{1}{2} \sup_{\mathcal{A}, \mathcal{B}} \sum_{\substack{A \in \mathcal{A} \\ B \in \mathcal{B}}} P(X \in A, Y \in B) - P(X \in A)P(Y \in B)$$

where \mathcal{A}, \mathcal{B} are partitions, which can be used to define asymptotic (or “local”) analogues of the assumptions of “indentically distributed” or “stationary”.

It is always a good idea to preprocess your data, either by sphering it or looking at its copula (it works fine with financial data).

Densities can often be well approximated by simple density classes, such as mixtures, products (as in independent component analysis) or copulas (the tail plots can help choose an adequate family of copulas).

Density estimation algorithms can benefit from efficient data structures, such as those used in the video game industry: for instance, an *evaluation tree* is a binary search tree that represents a setwise constant function, often enhanced by bounding boxes; it can be used to store a density estimate.

The third part of the book presents density estimators. Local averaging estimators include histograms; average shifted histograms (ASH), which approximate a kernel estimator with a triangular kernel; kernel estimators. You can use volume plots to check that the density

estimation is correct – even in high dimensions, when you cannot really “plot” it.

Series estimators include Fourier series, wavelets, projections on a lower-dimensional subspace, including shrinkage estimators, which also tweak the coefficients – the projection is not orthogonal.

Those estimators often minimize some empirical risk (MLE, etc.) and reduce the problem to a finite (combinatorial) one: choose the best density among a finite (huge) set of candidates (δ -net), or a lower-dimensional subspace, or a (countable, infinite) dense set in a lower-dimensional subspace (unless direct computation is possible, these are usually approximated by increasingly finer finite sets)

A greedy **adaptive density estimator** starts with a uniform density on a rectangle, splits it in two in the best possible way (to minimize some measure of empirical risk), and iterates. A **CART histogram** (a histogram is just a locally constant density) proceeds greedily, but also penalize for the complexity of the density, and prunes the result. **Dyadic histograms** only split the rectangles in their middle: they are actually Haar wavelet decompositions. Bootstrap aggregation, *i.e.*, averaging the density estimates on several bootstrap samples, can improve the result.

Stagewise minimization models the data as a mixture, but increases the penalty as more terms are added.

Boosting can be interpreted as a gradient search.

Bayesian methods in finance
S.T. Rachev et al.
Wiley (2008)

(The book is printed on low quality, nose-irritating, dust-mite-friendly paper.)

Clear and detailed introduction to bayesian statistics and its applications to portfolio management – nothing new.

Mean-variance portfolio allocation instability problems can be fixed by robust methods, portfolio resampling or bayesian methods. Bayesian methods take a non-informative or informative prior, mix it with the sample data, and output a posterior distribution of expected returns and variance matrix; in the non-informative prior case, the returns are the sample returns, while in the informative prior case they are shrunk towards the prior; in the non-informative prior case, the variance matrix is scaled *up*; for gaussian returns, the posterior distribution is a Student distribution.

More generally, instead of a gaussian model $N(\mu, \Sigma)$, one can “bayesianize” a regression – in an asset management context, regressions are called **asset pricing models** and the main examples are the CAPM ($r \sim f_1$, where f_1 are the market returns), the Fama-French 3-factor model ($r \sim f_1 + f_2 + f_3$ where the f_i are the market excess returns, the returns difference between large and small caps, and the returns difference between high and low price-to-book stocks), and the arbitrage pricing

ing theory (APT, $r \sim f_1 + \dots + f_n$, where you can freely choose the f_i). You end up with a posterior distribution for the regression coefficients. You can blend several models with bayesian model averaging (BMA).

The **Black-Litterman** framework allows an investor to start with an asset pricing model (e.g., the CAPM, in particular the expected return of all assets is the same) and progressively incorporate more information, in the form of forecasts on the returns of assets or portfolios of assets, each with a confidence level.

A market is said to be **efficient** if one cannot predict future price changes from past prices (weak efficiency), from current public information (semistrong efficiency) or from current public or private information (strong efficiency). In other words, tests of efficiency compare the following models.

$$\begin{aligned}\text{returns}_t &\sim \text{factors}_t \\ \text{returns}_t &\sim \text{factors}_t + \text{information}_{t-1}\end{aligned}$$

Economists often use a 2-step procedure: first estimate the beta of each asset, using time series regressions $r_{i,t} \sim f_t$, then estimate a cross-sectional regression, (*i.e.*, for a single date t) and test if the intercept is zero. In a bayesian setup, you can look at the posterior distribution of this intercept.

But this two 2-step process is incorrect (this is the “error in variables” problem). Instead, one can look for consequences of the efficient market hypothesis (EMH): for instance, under the CAPM, the market portfolio (or your benchmark portfolio) should be efficient (under the APT, some combination of the factor portfolios should be efficient). The tests then consider some measure of inefficiency, such as the difference in returns between the benchmark and the efficient portfolio with the same risk (vertical distance to the efficient frontier), or the difference in risk-adjusted returns μ/σ (Sharpe ratio) between the Benchmark and the tangent portfolio (a kind of “angular distance” to the efficient frontier) or the difference in certainty-equivalent returns (the returns of a risk-free portfolio with the same utility, *i.e.*, the difference in utility converted into returns). In a bayesian context, one can sample from the posterior distribution of the inefficiency measure (it is not tractable: you have to use simulations). Adding constraints (they change the efficient frontier) is straightforward.

One can also directly test for returns predictability,

$$\begin{aligned}\text{return}_t &\sim \text{factor}_{t-t} \\ \text{factor}_t &\sim \text{factor}_{t-1}.\end{aligned}$$

In a bayesian setup, one can estimate the posterior distribution of expected returns, and look at how it changes as the horizon increases: the impact of estimation uncertainty grows, making stocks less and less attractive. [Isn’t this in contradiction with the received wisdom that stocks are good for the long term and bonds better for the short term?]

The *Bayes factor* is not mentioned.

The estimation of **stochastic volatility** (SV) models, contrary to GARCH models, is not straightforward. Approximations, such as the quasi-maximum likelihood, or biased methods, such as the method of moments, are not reliable. The **efficient method of moments** uses an auxiliary, easy-to-fit model, such as the GARCH model:

- You have a complicated model $\mathcal{M}(\theta)$, e.g., a stochastic volatility model;
- You have a less satisfactory but tractable model $\mathcal{M}_0(\zeta)$;
- Estimate the parameter $\hat{\zeta}$ of the tractable model \mathcal{M}_0 on the data;
- For a value θ of the parameters of the complicated model, simulate data and estimate the parameters $\hat{\zeta}_\theta$ of the tractable model on the simulated data;
- Minimize $\|\hat{\zeta} - \hat{\zeta}_\theta\|$ (you can use a maximum-likelihood-based distance instead).

The fact that the sum of the GARCH parameters $\alpha + \beta$ is often almost 1 (it should be less than one for the model to be stationary and anything you do with it meaningful) could be an artefact of a misspecified model, e.g., time-dependent parameters. The linear model with student GARCH noise can be estimated via Gibbs sampling and can easily be generalized to a Markov regime-switching model (MSGARCH), by adding latent variables.

The stochastic volatility models are also latent-variable models: in this case, the number of unknown parameters (the model parameters and the unobserved volatility) is of the same order as the sample. They can be fitted by a Gibbs single-move sampler (*i.e.*, one updates a single parameter or non-observed variable at a time), but this converges very slowly if there are autocorrelations between the latent variables (and there are in the case of time series). This can be remediated by Kalman-filter-based **multimove samplers**.

The models can be extended to allow for jumps: add Bernoulli variables q_t to denote the presence of a jump and log-normal variables j_t for the amplitude of the jump; you can add jumps in the volatility as well: there are then fewer jumps and they are easier to interpret.

The distribution of asset returns are not gaussian: one can model them with a mixture of gaussians, an asymmetric Student T distribution, a stable distribution, an extreme value distribution (EVD), a skew-gaussian distribution (mixture of a gaussian and a truncated gaussian). Copulas are mentioned and the authors are aware of the need for and lack of tools to handle and estimate copulas in high dimensions; they can be used to generalize the Black-Litterman framework (copula opinion pooling). **Coskewness** (third moment between a stock and the market or a benchmark) or alternative measures of risk (expected shortfall (ES), aka conditional value at risk (CVaR)) can be incorporated into portfolio optimization.

A **risk model** is a concise way of describing the dependency between stock returns: it is sometimes written

as a linear model,

$$\mathbf{r} = \boldsymbol{\alpha} + \mathbf{f}\mathbf{e} + \text{noise},$$

where \mathbf{f} are the factor returns, sometimes as a decomposition of the variance matrix of returns V into factor exposures e and factor variance v : $V = \mathbf{e}\mathbf{v}\mathbf{e}'$. Bayesian methods can be used when estimating risk models. The risk or variance of a stock can be decomposed into the contribution of each factor (and their interactions); this can be generalized to other measures of risk.

***Simulation and inference
for stochastic differential equations***
**S.M. Iacus
Springer (2008)**

The first chapter of the book reviews stochastic processes and stochastic differential equations (SDE):

- Random number generation (Mersenne Twister);
- Monte Carlo simulations : the convergence of $\frac{1}{n} \sum g(x_i)$ to $Eg(X)$ is very slow, in \sqrt{n} , but is independent of the smoothness of g ;
- Variance reduction: preferential sampling, antithetic sampling (low discrepancy sequences, *i.e.*, quasi-random numbers, are not covered), control variables (find any relation between the quantity of interest and any other quantity, easier/faster to compute, for instance the call-put parity);
- (Total) variation, quadratic variation

$$[X, X]_t = \limsup_{0=t_0 < \dots < t_N=t} |X(t_{k+1}) - X(t_k)|;$$

- Conditional expectation;
- Martingales;
- Brownian motion, geometric brownian motion, brownian bridge (to simulate a brownian motion);
- The **Karhunen-Loève** expansion of the brownian motion expresses it as a *countable* sum:

$$W(t)(\omega) = \sum_{i=0}^{\infty} Z_i(\omega) \phi_i(t)$$

$$\phi_i(t) = \frac{2\sqrt{2T}}{(2i+1)\pi} \sin \frac{(2i+1)\pi t}{2T}$$

$$Z_i \sim N(0, 1) \text{ iid};$$

- Diffusion processes, *i.e.*, SDEs of the form

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t,$$

their stationary distribution, infinitesimal generator (only the formula is given, we are not told what it is nor why it is called like that);

- Ito's formula;
- If the diffusion coefficient σ of a SDE only depends on the state variable X_t (and not on t), the **Lamberti transform** turns it, via the Ito formula, into a SDE with unit diffusion coefficient.
- **Girsanov's theorem** is a change-of-measure theorem, that can be used to compute a Radon-Nikodým derivative, *i.e.*, a likelihood ratio.

This chapter ends with a list of (17) useful families of stochastic processes (Ornstein-Uhlenbeck (aka Vasicek), etc.) with unproven formulas (explicit solution, invariant law X_∞ , covariance function $\text{Cov}(X_t, X_s)$, conditional covariance function $\text{Cov}(X_t, X_s | X_0 = x_0)$, conditional law $X_t | X_0 = x_0$, etc.) and properties.

- The **Jacobi diffusion process**

$$dX_t = -\theta \left(X_t - \frac{1}{2} \right) dt + \sqrt{\theta X_t (1 - X_t)} dW_t$$

is reminiscent of copulas: its invariant distribution is uniform $U(0, 1)$; in particular, you can use it to build a process with a prescribed invariant distribution F as $Y_t = F^{-1}(X_t)$.

- Social scientists often consider feedback models (for population dynamics, political polarization, etc.), *i.e.*, models with a drift of the form $b(x) = r(\theta - x)$, whose diffusion coefficient σ^2 is constant, linear or polynomial in x , leading to a gaussian, Gamma or Beta stationary distribution.
- Epidemiologists often consider ordinary differential equations (ODE), to which you can just add a noise term $\sigma(X_t)dW_t$ to get a SDE.

The second chapter examines numerical simulations. The quality of a simulation Y_δ (where δ is the step size) of Y is assessed by looking at how the distance $E|Y_\delta(t) - T(t)|$ changes, for t fixed and δ getting smaller – $O(\delta^{1/2})$ is passable, $O(\delta)$ is better.

The **Euler scheme** is the naive discretization of the SDE: replace the d signs with Δ signs. Note that the discrete paths you get are not discretizations of paths you would get from the SDE itself: except for the brownian motion itself, there is a bias, you are sampling from a distribution of discrete paths close to but different from the desired one. For instance, the paths of the Cox-Ingersoll-Ross model

$$dX_t = (\theta_1 - \theta_2 X_t)dt + \theta_3 \sqrt{X_t} dW_t$$

are (a.s.) positive and therefore well-defined (if $2\theta_1 > \theta_3^2$), but those of the corresponding Euler scheme is not.

The **Milstein scheme** uses a second order Taylor expansion (via Ito's theorem).

$$Y_{i+1} = Y_i + b\Delta t + \sigma\Delta W_t + \frac{1}{2}\sigma\sigma_x((\Delta W_t)^2 - \Delta t)$$

If the diffusion coefficient σ only depends on the state variable X_t (and not on t), you can remove it with the Lamberti transform; things then magically improve: the (simpler) Euler scheme of the transformed process is the Milstein scheme of the initial one; in particular, its order of convergence is 1.

Higher order Taylor expansions (KPS scheme, second Milstein scheme) further improve the convergence; they are comparable to the Runge-Kutta method for ODE.

In some rare cases, the conditional distribution $X_t | X_s$ is known and exact simulation is possible: for the brownian motion, it coincides with the Euler scheme, but

it is also available for the geometric brownian motion (GBM, aka Black-and-Scholes model), the Ornstein-Uhlenbeck (Vasicek) process and the CIR process.

The Euler scheme replaces the SDE by one whose drift and diffusion are constant (this approximation is constant between any two consecutive steps, but it changes at each step): instead, one can approximate the drift by a linear function of X : the **Ozaki** (and Shoji-Ozaki, if the drift also depends on time) method uses such a local linearization.

Diffusion bridges can be simulated for ergodic processes (*i.e.*, those with no “arrow of time”): simulate two paths, one forwards, one backwards, and combine them if/when they intersect.

Exact sampling via rejection sampling is alluded to; it can be extended to diffusion bridges.

The third chapter addresses the estimation problem: you have a discretized sample path from a (time-homogeneous) diffusion process, you know (assume) it belongs to some family of SDEs (e.g., geometric brownian motion), and you want to estimate the corresponding parameters. As often, we will only have asymptotic results, in various setups: large sample (Δ constant, $T \rightarrow \infty$); high frequency ($\Delta \rightarrow 0$, T constant); rapidly increasing ($\Delta \rightarrow 0$, $T \rightarrow \infty$).

In real situations, even though your sample path looks fine (two or three parameters, hundreds of observations, the path looks smooth, sufficiently rich to present (not one but) many “patterns” that numerologists/chartists would be eager to interpret) there is not enough data and the asymptotic results are not valid.

In the ideal ergodic case, if we have a complete (continuous, not discretized) path, we know the invariant distribution; the (parameters intervening in the) diffusion coefficient can be estimated from the quadratic variation

$$\langle X, X \rangle_t = \int_0^t \sigma^2(X_s, \theta) ds;$$

the remaining parameters can be estimated by maximizing the log likelihood (the likelihood ratio is the Radon-Nikodým derivative)

$$\int \frac{b}{\sigma^2} dX_s - \frac{1}{2} \int \frac{b^2}{\sigma^2} ds.$$

In the discrete case, you could discretize those continuous estimators (they are not that efficient).

You can obtain discrete estimators directly, by replacing the integral by a sum – but you have to know the transition probabilities: it works for the Ornstein-Uhlenbeck model, the geometric brownian motion, the CIR model.

If the transition probabilities are not known, use one of the discrete schemes used to simulate trajectories of the SDE (Euler, Milstein, KPS, Ozaki, etc.): their transition probabilities are known. If $\Delta \ll 1$ and $N\Delta \gg 1$, then the Euler scheme is sufficient, if not, then none of those **pseudo likelihoods** is satisfactory.

The likelihood of $x_{t+\Delta}|x_t$ can also be estimated by simulating (with the Euler scheme and a step size δ smaller than Δ) trajectories starting at x_t .

Various expansions (Taylor, Hermite polynomials, etc.) of the likelihood can also be used, leading to **approximated likelihood** estimators.

Bayesian methods (MCMC simulations) can also be used.

Estimating functions (aka moments) are functions f such that $E f(X, 0) = 0$ iff $\theta = \theta_0$. If you choose the function wisely (e.g., apply the **infinitesimal generator** of the diffusion to any function h), this is analytically tractable and often consistent and asymptotically gaussian. You can also use the first and second moments of the transition density: the resulting estimators are consistent and robust to model misspecification.

The **generalized method of moments** is almost identical, but suggests you use more moments than needed: the model is overidentified.

The last chapter examines miscellaneous topics. The AIC (the log-likelihood, corrected for the dimension of the parameter space) could be used to compare models – but beware: it does not work.

Diffusions are amenable to non-parametric estimation. For instance, you can estimate the stationary density (with a usual kernel estimator), estimate the drift parameters (if this part of the process is parametric) and use the Kolmogorov equations, which link stationary distribution, drift and diffusion, to estimate the (non-parametric) diffusion. In a high-frequency setup, you can estimate the drift and diffusion as Nadaraya-Watson kernel regression estimators

$$b(x) = \lim_{t \rightarrow 0} \frac{1}{t} E[X_t - x | X_0 = x]$$

$$\sigma(x)^2 = \lim_{t \rightarrow 0} \frac{1}{t} E[(X_t - x)^2 | X_0 = x].$$

Change-point models can also be estimated with the Euler scheme.

The book is interesting but unpolished:

- Some formulas are wrong (not misleading, just meaningless); others are badly typeset, for instance, the limit in probability is written “ p minus lim”: why not use “plim” or “p-lim” (with a hyphen and consistent typeface) instead?
- The R code, most of which is available in the **sde** package (check the **sde.sim**, **DBridge**, **dcEuler**, **ksmooth**, **cpoint** functions), tends to return invisible objects: invisible objects are useful for functions called for their side effects and whose return values are rarely used, such as plotting functions; but if the return value is just too complicated or long to be printed, it is less misleading for the end-user to implement a **print** method;
- Some theoretical notions are missing or confusing:

the author talks of weak and strong solutions of a SDE but never defined them; the definitions of strong and weak order of convergence look fishy (the weak seems stronger than the strong).

Beautiful Data
T. Segaran et al.
O'Reilly (2009)

Empty.

Recent advances in harmony search
Z.W.Geem et al. (2008)

Harmony search (HS) is a music-inspired optimization algorithm, comparable to genetic algorithms, but it does not assume the genes are 1-dimensional: choose the value of each variable independently (no cross-overs, genes are not linear), either at random or from the *harmony memory*, *i.e.*, the current population (you can end up with more than two parents), and apply some *pitch adjustment* (mutation). This can be refined by putting back some dependency between the variables (ensemble HS (EHS)) and/or progressively changing the parameters (e.g., lower the mutation probability, as in simulated annealing: improved HS (IHS)).

There have been some applications in music (organum composition) and engineering (designing various structures, such as geodesic domes).

*AI methods for algorithmic composition:
a survey, a critical view and future prospects*
G. Papadopoulos and G. Wiggins

The following classes of composition algorithms have been investigated:

- Stochastic processes, such as Mozart's dice game (a musical analogue of R. Queneau's *Cent mille milliards de poèmes*), Markov chains (this requires a corpus, in which melody and rythm are often separated); artificial neural networks, machine learning, stochastic grammars (which try to reflect large structures but fail to account for "ambiguity" or the 2-dimensional structure of music – this is also my main criticism of Haskore: musical structures cannot be represented by a tree, we need a directed (but still acyclic) graph);
- Non-linear systems, iterated functions (but there is no way to evaluate the quality of the result);
- Knowledge-based systems: constraint satisfaction problems (CSP);
- Genetic algorithms (GA) (since it is almost impossible to design a good fitness function, you end up using humans: interactive GA).

Few of those use a geometric model of *pitch space* (pitches can be arranged in a *lattice*, e.g., with thirds in one direction and fifths in the other).

*Frankensteinian methods
for evolutionary music composition*
P.M. Todd and G.M.Werner

One can also consider **coevolutionary algorithms**, with two populations, of composers and critics.

Group Theory and SAGE: a primer
R.A.Beezer (2009)

SAGE is a common interface to several computer algebra systems (CAS) (Gap for group theory, Maxima for symbolic computations, etc.) but fails to follow the evolution of the components – that would be easy if they had an extensive test suite, such as all the computations in one of the many online documentstions.

For finite groups (mainly subgroups of \mathfrak{S}_n for $n \leq 50$), it looked fine, but symbolic computations (Maxima) and plotting (MatPlotLib) are broken (this may be Ubuntu-specific).

*Adventures in group theory:
Rubik's cube, Merlin's machine
and other mathematical toys*
D. Joyner (2008)

Applied group theory with Sage.

To solve the Rubik's cube (many other puzzles are covered): try to find sequences of moves (of the form aba^{-1} (conjugation by internal automorphism), $aba^{-1}b^{-1}$ (commutator), $(ab)^n$) that keep a lot of the facets unchanged; use them to first put the corners in the right places, then the edges, then the corners in the right orientations, then the edges. This can be formalized as a decomposition $g_0 = g_1g_2g_3g_4$ where $g_0 \in G$ is the permutation representing the current configuration;

$$\{1\} = G_4 \subset G_3 \subset G_2 \subset G_1 \subset G;$$

G_1 is the subgroup that fixes the corner positions; G_2 , the subgroup that fixes the corner and edge positions; G_3 , the subgroup that fixes the corner and edge positions, and the corner orientations;

$$\begin{aligned} g_0 &\in g_1G_1 \\ g_1^{-1}g_0 &\in g_2G_2 \\ g_2^{-1}g_1^{-1}g_0 &\in g_3G_3 \\ g_3^{-1}g_2^{-1}g_1^{-1}g_0 &\in g_4G_4. \end{aligned}$$

The Merlin machine (an $n \times m$ array of buttons with lights; pressing a button switches its light and that of its neighbours; your goal is to switch all lights on (or off) from an arbitrary initial position) is a matrix inversion problem: the initial state b is a vector of size nm (with coefficients in $\mathbf{Z}/2\mathbf{Z}$); the possible moves are also vectors of size nm , which can be arranged into a matrix A ; solving the puzzle is equivalent to solving $Ax = b$ (this is not always possible: A need not be invertible).

The **Cayley graph** of a group with prescribed generators $G = \langle g_1, \dots, g_n \rangle$ has the elements of G as vertices

and an edge $x \rightarrow y$ if $y = g_i x$ for some generator g_i . A solution to a puzzle is a path in the Cayley graph of its group. The diameter of the Cayley graph gives (a bound on) the minimum number of moves required to solve the puzzle.

Differential calculus and Sage **W. Granville and D. Joyner**

Sage relies on Maxima, which is woefully unstable (but Maxima is written in Lisp: how can they get a segmentation fault in Lisp?).

Sage notebooks are web-based equivalents of Maple's workbooks: you can use \LaTeX for mathematical expressions; they rely on Twisted for the access from a web browser.

Sage functions can be written in Python (but, of course, Python function cannot be symbolically differentiated).

Here is some sample code (press TAB after the name of a function such as `expand` or `simplify`):

```
x=var('x')
integral(x^2,x,0,1)
n(integral(x^2,x,0,1))
integrate(x^2,x)
a,b,c,d,e,f,x,y=var('a,b,c,d,e,f,x,y')
solve(ax^2+bx+x==0,x)
solve([ax+by+c==0,dx+ey+f==0],x,y)
tan?
diff(sin(x),x,4)
partial_fraction(1/(1-x^2),x)
limit(1/t,t=Infinity)
limit(1/t,t=0,dir="plus")
show(plot(x^2,-2,2))
maxima("sum(3*k+1,k,2,5)")
sum([f(i) for i in range(20)])
```

Sage for Newbies **T. Kosan (2008)**

Integral calculus and Sage **D. Hoffman et al. (2009)**

A calculus book relying on Sage: most of the code is actually Python and the pictures are awful (low resolution, lossily-compressed bitmaps, scaled by a different (random?) and ever changing factor in the horizontal and vertical directions).

Asymptote: the vector graphics language

Asymptote is a variant of Metapost (a language to describe and produce PDF figures to be included in \LaTeX files) with a C++-like syntax: semi-colons, parentheses around function arguments; standard data types (real, triple, vector, matrix, string); standard libraries (operations on strings, arrays, vectors, matrices – including mathematical operations such as matrix inversion or the Fast Fourier Transform (FFT)); plotting li-

braries (any kind of plot with axes, even with logarithmic or broken scales, 3-dimensional plots, diagrams, etc.); ability to include code directly in your \LaTeX files (in the `asy` environment; use the `asy` command to produce the plots). The labels are typeset in \LaTeX , with some Unicode if needed (cyrillic fonts, Asian languages through the old CJK package – no mention of \XeTeX). They seem to have kept all the good features of Metapost, *except* the constraint-solving capabilities – but given the emphasis on “scientific” (*i.e.*, numeric) plots, this may not be an issue.

For an extensive gallery of examples, including many showing that the constraint-solving feature was not that important, check <http://www.piprime.fr/asymptote>.

Primes in P **M. Agrawal et al.**

There exists a polynomial algorithm to determine whether a number is prime. The algorithm is very simple (but not that fast: $O((\log n)^{21/2})$) and based on a generalization of Fermat's little theorem:

$$a \text{ prime} \iff (X + a)^n \equiv X^n + a \pmod{n}$$

for $a \in \mathbf{Z}$, $n \in \mathbf{N}$, $n \geq 2$, $a \wedge n = 1$.

The complexity classes are the following:

- P: there exists a polynomial algorithm to determine if a number is prime
- co-NP: to prove that a number is not prime, it suffices to provide a “non-primality certificate” (the decomposition into a product of prime factors); the verification (product) can be done in polynomial time;
- NP: to prove that a number is prime, it suffices to provide a certificate from which one can check that the number is indeed prime (since Prime is P, the certificate can be empty);
- NP-Complete: a problem is NP-complete if all NP problems can be reduced to it in polynomial time.

A comparison of random forest and its Gini importance with standard chemometric methods for the feature selection and classification of spectral data **B.H. Menze et al** **BMC Bioinformatics (2009)**

Regularized methods can tackle large numbers of predictors, but do not identify and discard irrelevant variables (variables are often called “features”): in chemometry, one often applies them after some feature selection algorithm, *e.g.*, with the *random forest Gini importance* (total change in entropy brought by a given variable or set of variables in a random forest).

Planet: massively parallel learning of tree ensembles with MapReduce **B. Panda et al.**

Classification trees or regression trees (a “common data mining task” at Google) can be parallelized.

***A course in credibility theory
and its applications***

**H. Bühlmann and A. Gisler
Springer Verlag (2005)**

On the use of bayesian models, or more generally hierarchical models, in actuarial science (*i.e.* “insurance”) – they use a different vocabulary: “credibility theory” means “hierarchical model”.

Dynamical bias in the coin toss

Coin tossing is not random: with probability 0.51, the coin lands as it started.

Coin spinning is even worse: depending on the age of the coin, the probability of tails can be as high as 0.9.

***Tag cloud drawing:
algorithms for cloud visualization***
O. Kaser and D. Lemire

Tag clouds, in most websites, are often just an alphabetic list of words of varying sizes, with no attention given to esthetics, ease of use (the alphabetic order is far from optimal) or space usage. Here are a few ideas to improve them:

- Keep the alphabetic order but use a decent (T_EX-like, dynamic-programming-based algorithm (compute the badness b_{ij} of a line containing words i to j) instead of the greedy, Word-like one) line-breaking algorithm; you may want to shuffle the words a dozen times and keep the best layout;
- Sort the tags by decreasing height and add them one by one, greedily, to the first available line; create a new line if needed;
- Take the relations between the tags into account (model them as a weighted graph) and use some heuristic from the *placement problem* in EDA (Electronic Design Automation), e.g., force-directed methods (view the graph edges as springs), simulated annealing, or heuristics for the (NP-hard) *min-cut placement* problem, which recursively cuts the set of tags in two (left/right or top/bottom) so that the cut size (total weight of the edges between the two parts; also add the influence of external edges) be small and the bipartition balanced; the result can be implemented with nested HTML tables (but it looks very tabular).

Using trees to depict a forest

**B. Liu and H.V. Jagadish
VLDB 2009**

When a database query returns too many results, instead of ranking them, present a “representative” subset, obtained by clustering.

Scientific database applications have the following requirements:

- Non-tabular data-types: n -dimensional arrays, ragged arrays, nested arrays, sequences, graphs, meshes;
- **Provenance** tracking, snapshots (“named versions”, *i.e.*, you can modify the data without tampering with other users’ work and without having to copy it), no-override storage manager (just an “update time” dimension);
- Grid storage (because of the volume of data: 100 terabytes);
- Ability to work on data without having to load it (HDF-5, NetCDF file formats);
- Database operations (join, etc.) should account for imprecision, e.g., by adding a “precision” to a column and considering values differing by less than this threshold as equal; more elaborate models could assume the data in a column is gaussian, estimate its mean and variance, and output results with a confidence interval;
- User-defined functions (e.g., in C++).

SciDB does not exist: it is just a wishlist.

***ConvergenceConcepts: an R package to
investigate various modes of convergence***

**P. Lafaye de Micheaux and B. Lique
R Journal (2009)**

If you have a sequence of random variables, or random number generators (or several samples – it may even work with a single path, if it is sufficiently “representative” and “ergodic”), you can study their convergence as follows:

- Plot several samples paths $(X_n)_n$ and check if they leave $[-\varepsilon, \varepsilon]$ for $n > N$: if most do not, then $X_n \rightarrow 0$ a.s.;
- Estimate the proportion of paths that leave this interval,

$$a_N = P[\exists n \geq N : |X_n| > \varepsilon],$$

if it converges towards zero (in most (textbook) examples, it is really clear-cut), the sequence converges almost surely;

- In the plot of the sample paths, instead of the $[N, +\infty[\times [-\varepsilon, \varepsilon]$ infinite rectangle, just consider the narrower $\{n\} \times [-\varepsilon, \varepsilon]$ rectangle: if the proportion of paths outside it

$$p_N = P[|x_N| > \varepsilon]$$

decreases to 0 as N increases, there is convergence in probability;

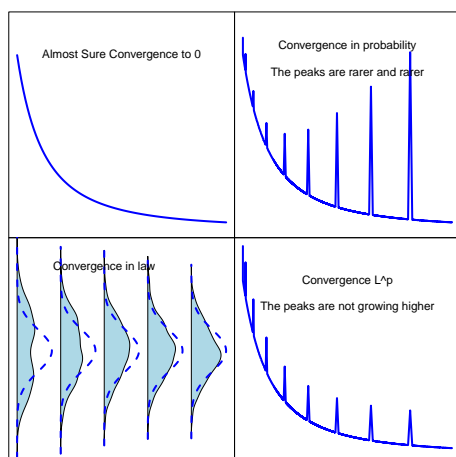
- For the convergence in law, you can plot the cumulative distribution function (cdf) and check if it moves towards the supposed limit; one could also use an animated quantile-quantile plot (or qqplot, or his-

togram, etc.) or a plot of the $|F_n(x) - F(x)| \sim x + n$ surface;

- For L^p convergence, just compute an estimator of $E|X_n - X|^r$,

$$\hat{e}_n = \frac{1}{|\Omega|} \sum_{\omega} |x_n(\omega) - x(\omega)|^r.$$

The GUI is ugly but interactive.



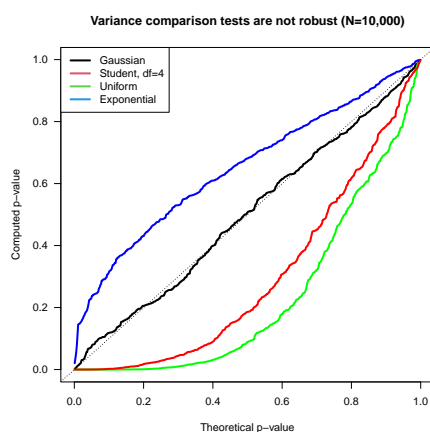
**Understanding convergence concepts:
a visual-minded and graphical
simulation-based approach**

P. Lafaye de Micheaux and B. Lique
The American Statistician (2009)

**asymptTest: a simple R package
for classical parametric statistical tests
and confidence intervals in large samples**

J.-F. Coeurjolly et al.
R Journal (2009)

Mean comparison tests are robust to non-gaussianity, but variance comparison tests (the 1-sample χ^2 variance test; or the 2-sample Fisher ratio test, `var.test`) are not, even for large samples.



Robust tests and confidence intervals can be obtained by considering asymptotically gaussian quantities such

as

$$\frac{S_n^2 - \sigma^2}{\hat{\sigma}_{S_n^2}}.$$

**copas: An R package
for fitting the Copas selection model**

J. Carpenter et al.
R Journal (2009)

The Copas model accounts for publication bias (more significant or smaller studies (with more extreme results) are more likely to be published) and is used to improve meta-analyses in medical studies.

Use R 2009 Conference

Examples

Year after year, there are fewer and fewer examples that boil down to a simple linear regression: they are replaced by non-linear (sometimes non-parametric) and/or hierarchical linear (or survival, logistic, Markov, spatial, even spatio-temporal multiscale (country/micro-region/region)) penalized models, sometimes with added boosting or a bayesian prior; some presenters use the bootstrap to have more robust confidence intervals.

Examples included

- Non-linear models in biochemistry, to estimate drug interactions, dose-response curves;
- Shape analysis;
- Text mining (for gene names in biology);
- Meta-analysis (aggregating several studies, e.g., with microarray data);
- Time series (building a business cycle (recession) indicator from NBER data: transform and/or filter the data, use non-linear and/or Markov switching (MS) models);
- Forecasting multivariate time series (to optimize a frozen goods supply chain or forecast electricity consumption);
- Multivariate analysis (with **FactoMineR**);
- Design of experiments (DOE), survey design (compute the sample size, in each stratum, for a given desired precision, to estimate fisheries revenues)
- Subject randomization system (subjects for a randomized trial arrive one by one; we then learn in which strata (gender, age, etc.) they are; we decide to give them treatment A or B by flipping a coin; we can hasten the convergence by keeping a running difference $n_A - n_B$ for each stratum and using a biased coin to redress any unbalance).

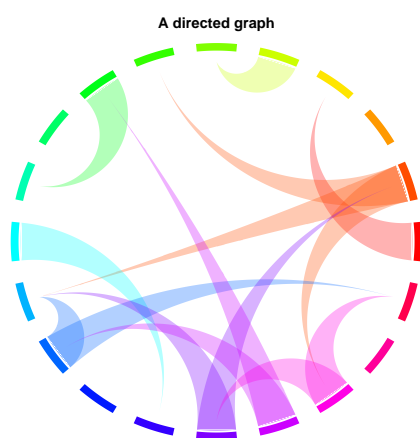
The large-scale ThomasCook example was interesting:

- They store the data in an SQLite table (modified to hold 30,000 columns – with PL/R, they had ended up writing R code that called PostgreSQL that called R: it was a nightmare to debug);
- They use R for ETL (Extract, Transform, Load), with some Python (BeautifulSoup) for screen-scraping;

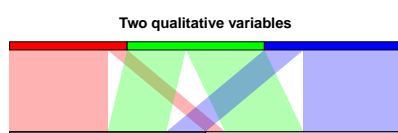
- `glmpath` to select variables;
- `randomForest` (on the selected variables) to predict the speed at which the planes will fill;
- The prices are optimized with the random forest model;
- They add a fuzzy inference engine (based on the `sets` package) to prevent cannibalization (you lose clients to yourself) and allow the business to add their input to the model;
- They provide a graphical interface with wx-Python/RPy2 (and `py2exe` to create Windows executables that do not require Python on the clients); R remained on a central server (some call it SaaS: Software as a Service).

Plots

Kaleidoscope plots are those impressive, dense, polar coordinate plots, judiciously using transparent colours, you sometimes see in bioinformatics, often produced by *Circos* (GPL, unrelated to R). There was one on the title page of *Beautiful code*, *Compelling Evidence*, which explains how to use Haskell and OpenGL to produce graphics. The basic idea stems from the graph layout problem: if you are completely clueless about how to plot a graph, just put the vertices in a circle; if the graph is weighted, draw the edges as varying-width ribbons;

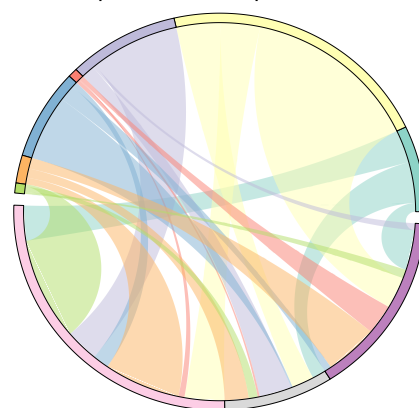


finally remark that a contingency table is a (bipartite: rows and columns) weighted graph



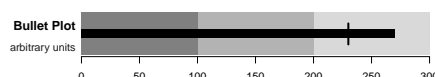
and draw it in polar coordinates (pay attention to the order of the segments and the ribbons, label the segments, add axes). Use it if you want artistic plots (that make the user wonder what the plot is about) or if your data is too large to be amenable to more traditional plots.

Two qualitative variables in polar coordinates

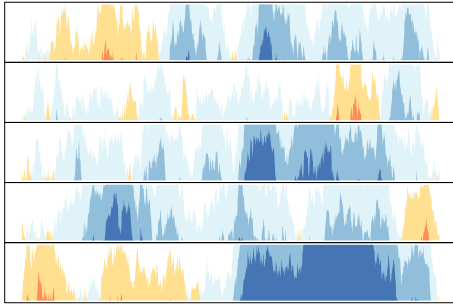


Visual analytics are statistical graphics made by artists or good communicators: they should capture the audience's attention and convey the speaker's ideas. The most salient such example is *GapMinder* (a flash-based application to display multivariate time series, as animations) and its narrative power (it helps you efficiently tell a story, but might not be the best tool to *discover* that story). As statisticians, we can try to reduce the information displayed to what is relevant and add model information (distribution, prior, expected values under H_0 , etc., for instance a map of cholera cases can be complemented by well positions and their Voronoi tessellation). R provides graphics for diagnostics, presentation (but no animations or interactivity), and not much for data exploration (*iplot*, *iplot eXtreme*, *ggobi*) – redundant functions do not help (for instance, you can draw a histogram in any of the following ways: `hist`, `truehist`, `histogram`, `qplot(...,geom="histogram")`, `iplot`).

An **information dashboard** (ID) presents all the information needed in a single, easy-to-read page; it can use bargraphs, stacked bargraphs, linegraphs, scatterplots, boxplots, sparklines, treemaps, **bullet graphs** (a thick horizontal line from the origin to represent a single value, with a tick for a reference point (target, last year's value, etc.), and a coloured background to represent how good the value is (3 values, from dark (bad) to light (good) grey; do not rely on colours); do not forget a textual label, with units (smaller), and a scale)



or **horizon plots** (to plot a time series, represent both positive and negative values above the axis, with a different color (blue/red), put the possible values into three bands, of darker colours, and overlay them) – no piecharts.



The `iPlot` package adds **interactive graphics** to R, but cannot be used for large datasets: there is no OpenGL in Java and the data is duplicated between R and the JVM. The soon-to-be-released `iPlot eXtreme`, in C++, should address those problems.

The `PairViz` package uses `graph` and `igraph` to order the variables in a **parallel coordinate plot** using graph-based algorithms (hamiltonian or eulerian path); this can be combined with `scagnostics` if there are many variables.

R can be linked to external visual exploratory analysis tool: `ggobi`, `Visplore` (opensource, but early alpha), etc.

To plot the results of a clustering,

```
bwplot( value ~ variable | cluster )
```

you can add a reference point (the global average of each variable), fine-tune the order of the clusters, of the variables, highlight important variables (or grey out unimportant ones – this will depend on the cluster).

Statistical Tests

Multiple testing often assumes the tests are independent and then controls the *family-wise error rate* (FWER, probability of having at least one wrongly rejected null hypothesis), the *false discovery rate* (FDR, proportion of wrongly rejected null hypotheses) or the *false non-discovery rate* (FNDR, proportion of wrongly non-rejected null hypotheses). If you simulate data similar to yours, you can have an idea of the distribution of the FDR and FNDR and see how it changes as the correlation of the data increases. It is possible to include some knowledge of the correlation structure into the testing procedure: these “factor-adjusted test statistics”, implemented in the `FAMT` package (factor analysis for multiple testing), can improve both FDR and NDR.

When estimating a *p-value* by Monte Carlo simulations (bootstrap, permutations, etc.), some people do not want the *p-value* but are happy with just knowing if $p \leq \alpha$ for some predefined threshold α . Because of this, we can do with slightly less data (than if we wanted p , i.e., all values of α) by looking at the path $(\sum_{1 \leq i \leq n} T_i)_{n \geq 1}$ of partial sums of the test statistic and checking if it remains in or leaves some box- or tunnel-shaped region (sequential Monte Carlo *p-values*).

The `car::Anova` function implements nested **linear tests** (type II tests, i.e., tests of $H_1|H_2$, where $H_1 \implies H_2$) and unconditional linear tests (type III tests) with an *F* statistic.

Matching

When comparing the two models $y \sim x$ and $y \sim \text{treatment} + x$, where the treatment is a binary variable, some people like to have two observations for each possible value of the covariates, one for each value of the treatment. This is usually not the case, but we can instead “fabricate” new observations: **matching** is very similar to regression, but it is not linear (it could be seen as a “non-parametric regression developed by non-statisticians”). This looks like an imputation (or inference with missing data) problem, but it is apparently addressed with ad hoc methods: coarsened exact matching (CEM) is one of them – there are 10 matching packages for R... An application to program evaluation (“program” means sanitation improvement, etc.) was presented.

Penalized Estimators

Penalized estimators (such as non-parametric regression) are good but biased by construction: you can try to estimate and reduce the bias; the `ibr` package implements such an **iterative bias reduction** – but there were too few details in the presentation.

Penalized regression minimizes $\text{RSS} + \sigma^2 k \lambda$, where RSS is a sum of squares, k is the model size, and λ the penalty for complicated models. AIC (Akaike information criterion) uses $\lambda = 2$ but overfits large datasets; BIC (bayesian information criterion) uses $\lambda = \log n$. **Adaptively penalized regression** minimizes $\text{RSS} + \sigma^2 \phi(k, n)$, where ϕ is motivated by the FDR (false discovery rate), i.e., the expected number of wrongly rejected H_0 in a multiple test (it has a higher power than the too conservative FWER (family-wise error rate), which controls the probability of one or more incorrectly rejected H_0 ; with the FDR, you do not have a single *p-value*, but a *sequence* (for the first, second, etc. test): $p/m, 2q/m, 3q/m$).

A full **regularization path** (i.e., a penalized estimator, for all values of λ : lasso, with its L^1 penalty; ridge, with its L^2 penalty; elastic net, with both: $\frac{1}{2}(1 - \alpha) |\beta|^2 + \alpha |\beta|$) can be efficiently computed by **coordinate descent** (discretize the path, estimate one coordinate at a time until convergence, repeat for the next point on the path, etc.). Coordinate descent should work any time your optimization is of the form something + $\lambda \times \text{penalty}$, for instance to build undirected graphical models (it gives you a sequence of graphs) or to complete a matrix (e.g., a movies \times rates matrix, not observed completely).

In **credit rating**, to estimate the probability of default, you can replace the (linear) logit model by a generalized additive model (GAM). (There are (too) many such estimators in R, not only `gam` and `mgcv`; if you hesitate, use `gam` (more stable for small datasets),

unless you have a lot of relatively clean data.)

The **Influence.ME** package provides regression diagnostics (influential observations, etc.) for mixed models; the **merBoot** package uses bootstrap to provide empirical p -values for mixed models.

Non-linear models

The **nlstools** package provides a few functions to help fit non-linear models: **preview** to help you (graphically) choose the starting values; **nls** to fit the data; **overview** as an alternative to **summary**; **plotfit** to display the data and the fitted curve – also check **nlsResiduals**, **test.nlsResiduals**, **nlsContourRSS** (to identify ill-conditioning), **nlsConfRegions**, **nlsJack**, **nlsBoot**.

Univariate data

To estimate the **mode** of unimodal data, you can compute density estimators with wider and wider kernels, look at their local maxima and remember their positions when they disappear: you get a dendrogram-like plot – this can actually be generalized to higher dimensional data (apply some dimension reduction algorithm afterwards, if needed): check the **denpro** and **delt** packages.

The **R2lUniv** package provides graphical summaries of multivariate data, in \LaTeX , but is not as nice and compact as **Hmisc::describe**.

The **DTDA** package provides functions to analyze truncated data (estimator of the cumulated distribution function (cdf), etc.).

The **fitdistrplus** package helps you fit distributions (as **MASS::fitdistr**), but allows for *censored* observations, provides skewness/kurtosis plots (with the dataset, bootstrap replications, and the regions attainable by the models you are considering) and goodness-of-fit tests (Anderson-Darling test to compare the data and the fitted model – it accounts for estimated parameters).

The Tobit model

$$y_1 = \beta x + \varepsilon \\ y = y_1 \cdot \mathbf{1}_{y_1 > 0}$$

can be generalized to **multiple hurdle** models

$$\mathbf{y} = \beta \mathbf{x} + \varepsilon \\ \varepsilon \sim N(\mathbf{0}, V) \\ y = y_1 \cdot \mathbf{1}_{y_1 > 0, y_2 > 0, \dots, y_m > 0}$$

with the **mhurdle** package.

The **mlogit** package fits multinomial logit models (wasn't there already an implementation in **nnet::multinom**?).

Time Series

Markov Switching models can be fitted by expectation maximization (EM): estimate the state at each time; then the parameters in each state; iterate.

You can **mine** a large number of time series by computing a few metrics for each of them (average, volatility, minimum and maximum of the derivative, etc. – use the methods of *functional data analysis* (FDA)) and plot the resulting multivariate dataset (1- or 2-dimensional plots, PCA (principal component analysis), etc.). This is the idea behind **scagnostics**, applied to time series.

In a *threshold autoregressive model* (TAR) model, the autoregression (AR) coefficient depends on the previous value (often, one allows two values depending on the sign; sometimes three values corresponding to “significantly positive”, “significantly negative”, “small”). The idea can be generalized to **threshold cointegration**: two time series are threshold cointegrated if they have a threshold stationary linear combination y , *i.e.*, $y_{t+1} = \alpha(y_t)y_t + \varepsilon_{t+1}$, with $|\alpha(y_t)| < 1$ if $|y_t|$ is large.

The Kalman filter can be robustified to resist to either outliers or regime changes (or even both, but with a delay) with the **rLS** and **ACM** algorithms (not detailed enough).

With the **seewave** package, R can process sound samples (the plots are nice and the PDF presentation even contained an animation) – of course, since R tends to store data in memory, this is only limited to sound *samples*.

Breakpoint models are used in bioinformatics to model the ratio

$$\frac{\text{number of copies of a gene}}{\text{number of copies in the reference}}$$

as a locally constant function of the position on the chromosome; the breakpoints can be estimated with *dynamic programming* – could dynamic programming be applied to other breakpoint problems? I rarely see it outside bioinformatics.

The usual **RMetrics** presentation(s) focused on date arithmetics (the **timeDate** and **timeSeries** packages are aware of weekdays, businessdays, for a given financial center – but I did not really understand their explanation of the **recordIDs** slot) and *portfolio optimization*.

Spatial Models and GIS

The **GeoXp** package provides a few plots for spatial data: *driftmap* (not defined), *angle map*,

$$f(x_1) - f(x_2) \sim \text{angle}(x_2 - x_1),$$

where the angle is with respect to the horizontal axis), *neighbourhood map* (number of neighbours versus distance to the nearest neighbour), *Moran plot* (to check for spatial autocorrelation, plot the value of the variable of interest at a point versus the average value in a neighborhood of this point – you could also use two different variables), outliers (Mahalanobis distance between pairs versus Euclidian distance between pairs); those plots could be linked to a map.

Multivariate Data

To the list of *multidimensional scaling* algorithms, `cmdscale`, `isoMDS`, `sammon`, you can add `isomap`, which tries to find a (non-linear) 2-dimensional submanifold, smooth but close to the data. As always, try all algorithms and try to replace the dissimilarities by their ranks (it does not matter if the matrix is not metric).

The `biclust` package implements many **biclustering** algorithms (for binary, quantitative, ordinal or qualitative variables).

Hierarchical clustering can be performed on principal components (some claim that discarding the last factors of a factor analysis makes the clustering more robust, others claim the opposite); the clusters can be defined by representative individuals (actual data points), variables (as in biclustering) or axes (as in a biplot).

fMRI data can be analyzed through spatial and even (spatio-)temporal ICA (independent component analysis) if you have a sparse description of the singular value decomposition (SVD); also check the `PTAk` package for principal tensor analysis – a similar idea was presented a few years ago at SigGraph.

Canonical correspondance analysis (CCA) and *partial least squares* (PLS) are very similar: the former looks for linear combinations $\alpha'X$ and $\beta'Y$ of X and Y that maximize $\text{Cor}(\alpha'X, \beta'Y)$, the latter maximizes $\text{Cov}(\alpha'X, \beta'Y)$. Both can be regularized: the ridge (replace $(XX')^{-1}$ with $(XX' - \lambda I)^{-1}$) is popular with CCA, the lasso (an L^1 penalty) with PLS.

Data envelopment analysis (DEA, *i.e.*, the construction of *efficient frontiers* in the input \times output or risk \times reward space) can use CCA as a first step, to select the variables to use; beware of outliers: DEA focuses on extremes. DEA can also be applied to medicine.

Distributed quantile estimation (compute a few quantiles on each client, convert them into an estimated cumulated distribution function (cdf), send it to the server which then averages those cdfs) can be generalized to higher dimensions with PCA (only compute the quantiles on the first principal components); this idea can be applied to *monitoring* (e.g., to spot when a coffee machine is about to run out of coffee by listening to it).

Multiple tests and variable selection can be improved by accounting for correlation (with a correlation-adjusted T score). For instance, in genetics, you can group genes in the same pathway (or, if you do not have this information, in the same correlation neighbourhood). For feature selection, you should control the FNDR (false non-discovery rate) rather than the FDR (false discovery rate).

Several presentations focused on **multiple imputation**: exploring/visualizing the structure of missing values with the `VIM` package (there is also a GUI); fixing the separation problem (more variables than observations would remove all the randomness from the impu-

tation procedure) by adding a bayesian prior with the `mi` package; reducing the bias in variance estimates after multiple imputation with a misspecified model with “estimating equations” (no details).

Trees

The `partykit` package unifies the various tree packages (`rpart`, `knnTree`, `Rweka`, `party`, `randomForest`, `gbm`, `mboost`) and should facilitate the implementation of other algorithms; it also provides a PMML interface (PMML is an XML file format to exchange statistical models between applications, mainly used in data mining).

Random forests are not only used for *prediction* or *feature selection*, but also to measure how important each variable is; contrary to univariate measures, they can take interactions into account. The **permutation importance** of X_j is the average decrease in classification accuracy after permuting X_j . This is problematic because a variable can appear important because it is independent from the variable to predict, or just from the other variables. Conditional permutation importance (implemented in the `party` package) can help focus on the former.

The standard benchmark to measure the performance of a tree algorithm is C4.5 – its licence does not allow you to use it, but it has been reimplemented as J4.8 in Weka, available through `Rweka`. Other algorithms include: CART (classification and regression trees) in the `rpart` package; unbiased recursive partitioning (QUEST in the `LohTools` package) and conditional inference trees (CTree in the `party` package). They all perform well, except C4.5/J4.8 without cross validation (cross-validation improves *all* those algorithms and is already included in some of them: if not, do not forget to add it).

Classification trees (CART, in the `rpart` package) can be generalized to a multivariate gaussian outcome (`longRPart`).

After a hierarchical classification, one usually cuts the dendrogram, but you do not have to do it horizontally – *permutation tests* can help you find a good, non-horizontal cut.

Graphs

While the *graphical model* R packages landscape is getting less empty, it remains more heterogeneous than other domains, and is still plagued with external, Windows-only and/or closed source solutions, rarely useable in a professional context: WinBugs, MIM, Genie/Smile (closed-source, but free to use, even commercially), etc. Here are a few examples besides the classical `gR` (actually a combination of several packages) and `gRbase`:

- The `ReBaStaBa` package describes an already constructed bayesian network (*bayesian networks* provide a concise description of a joint distribution);
- The `IdR` (influence diagrams in R) package, can ap-

parently learn the graph structure from the data – too few details in the presentation.

Graph (or link) data abound, from the web (links between Wikipedia articles) to linguistic databases. An **ontology** is a list of words, with links between them; contrary to a taxonomy, those relations are not limited to speciation/generalization but can be arbitrary complicated; polysemy and synonyms are often prohibited (to avoid the mess of gene and protein names); ontologies can be used to represent knowledge and make deductions (an ontology can be seen as a graph theorist’s reinvention of Prolog). OWL (Web Ontology Language) may appear powerful, but is not robust at all; good old RDF (resource description language: it lists triplets (subject, predicate, object), or equivalently (node₁, edge type, node₂), *i.e.*, describes a graph with coloured edges – contrary to OWL, it does not allow for constraints on the predicates such as cardinality, transitivity) is sufficient for most practical purposes.

The **igraph** and **pagerank** packages provide graph-theoretic algorithms; for instance, they can be used to create a **tag cloud** using multidimensional scaling (MDS) for the positions. You may also want to check Wordle (non-free, but free to use), which takes the shape of the words into account, or Many Eyes’s bubble chart (apparently no licence), which uses a circle packing algorithm and amounts to a square root transform.

The **sna** package (social networks) can be combined with the **tm** package (text mining) to analyze mailing lists.

The result of a centroid-based clustering algorithm (*k*-means, PAM (partition around medoids), etc.) can be assessed with:

- The *silhouette plot*, $d(x, c_2(x)) \sim d(x, c_1(x))$, where $c_1(x)$ and $c_2(x)$ are the centroids closest and second closest to observation x ;
- Topology-representing networks (TRN) or *neighbourhood graphs*, weighted graphs with the centroids as vertices and the weight between c_1 and c_2 is the number of points x such that $c_1(x) = c_1$ and $c_2(x) = c_2$ or the average mean average distance.

The **gcExplorer** package relies on **Rgraphviz**, **graph** and **symbols** (to add plots (barplots, etc.) in the graph vertices) to display these, interactively.

Machine Learning

Association rule mining can be seen as a form of “non-symmetric correspondance analysis” for binary data: it can rely on brute force or multiple correspondance analysis (MCA); some algorithms also use exogenous data (e.g., socio-demographic data for a client \times item dataset).

The **set** package provides (finite) set operations, that can easily be generalized to **fuzzy sets** (just replace the topos of truth values $\{0, 1\}$ by the interval $[0, 1]$); applications include relations (order, partial order,

equivalence, etc.) and how to average them (e.g., **voting systems**) – this leads to non trivial optimization problems.

The **TopkLists** package implements algorithms to average **ordinal data**, *i.e.*, rankings, when m raters provide the top k elements in a list of N (more voting systems).

Dynamical Systems

Surprisingly, R is sufficiently fast to simulate dynamical systems (e.g., in ecology), *i.e.*, large systems of differential equations; check the following packages: **desolve** (rather than **odesolve**, which could only tackle small/toy problems) for dynamical systems; **rootSolve** for the steady state solution; **linSolve** for least square solutions (when your linear system is underdetermined, just add $\text{Min } \|x\|^2$ to turn it into a quadratic program); **simecol** for applications in ecology.

High Performance Computing

bigmemory, which uses the Boost C++ interprocess library, paliates the lack of multithreading in R through shared memory or file-based matrices; it also provides iterators to hide (or make implicit) the parallelism provided by **multicore**, **snow**, **nws**. Also check **ff** (matrices limited to $2^{31} - 1$ elements), **ffdf** (as fast as bigmemory), **sqlite** (sufficiently fast), **BufferedMatrix** (inefficient), **filehash** (sufficiently fast).

RCpp allows (eases) the use of C++ and STL containers; conversely, **RInside** allows you to call R from C++; also check **rdyncall** to use C libraries from R.

Several packages provide some parallelism: **multicore**; **snowfall**, **nws**, **Rmpi**; **gridR** (with parallel computations, beware of random number generators: the random numbers on each node should look independent).

Threads (as opposed to parallelism) are needed to process real-time (streaming) data. R will remain mono-threaded for the foreseeable future; in the meantime, use several process (with some C/C++ where needed) and shared memory (with bigmemory) between processes.

R can be used in simple web applications (the user fills in a form and a plot is updated), thanks to the **RApache** module, with some Ajax (Prototype, Scriptaculous, YUI, jQuery, Dojo, Ext), XML or JSON (better than XML for large datasets), and **hwriter** (for static reports, I use **Sweave**).

RdWeb is a (still immature) web interface to parallel computing in R (**RApache** and **Rserve** do not focus on parallelism); it requires a batch system (**at**, openPBS, Torque, etc.)

A **task scheduler** is a hybrid between **make** and **at/crontab**: it automatically executes tasks after a given time, when some conditions (file availability, etc.) are met and allows tasks to be dependent on one another (you can write a quick and dirty task scheduler with **make** and **crontab**: from the crontab, launch **make -k -j 100** in a directory every minute; tasks simply

fail if their starting conditions are not met; track which tasks have run or are currently running by creating empty files with predefined names). The Coalition task manager (in Python, with Twisted Matrix, on Google-Code) handles dependencies, load balancing; it does not seem to provide repeated tasks (“every day”), or tasks starting after a given time. There is nothing specific to R (and there should not be): it just launches executables, which can be R scripts.

A **workflow engine** allows you to write (the equivalent of) Makefiles (*i.e.*, describe dependencies between tasks) graphically and run them on a grid, such as Knime, which focuses on *data pipelines*: it provides all the operations you can want on tables of data, the most common data mining algorithms and plots (Weka), and interfaces with R and BIRT (a J2EE reporting system).

The Nimrod toolkit combines a workflow system (Kepler), a design of experiments framework (DOE, more precisely “computer experiments”, whatever that means – simulations?) and **gaussian processes** (with the **mlegp** package).

Biocep-R is a javaesque application server: huge, exhaustive, unwieldy and fully buzzword-compliant.

The **hive** package uses Hadoop (a distributed computed framework, with a distributed file system, trying to run the computations near the data, initially developed by Yahoo and currently maintained by the Apache foundation) for text mining tasks (with the **tm** package).

Most of those cloud computing frameworks (Amazon EC2, etc.) can generate or use models in the PMML format (designed to exchange statistical models between applications).

GUI

JEdit can be used as an IDE for R (many people advise **Eclipse/StatET**): it interfaces with the **codetools** package (warnings are presented as spelling mistakes would be in a text editor); provides a tree view of the code (based on the R parser), completion popups (for colours or plotting symbols, you have the actual colour or symbol), an object browser, a debugger.

The building blocks for SciViews (an R GUI) are available as individual, reusable packages; some were used to build other GUIs such as Zoo/PhytoImage, an R-based graphical software for image analysis; it also uses the ImageJ image processing Java library (public domain).

Mayday is a visualization platform (in Java, interactive, aware of meta data, with the ability to write plugins) for numeric matrices; it now has an interactive R shell, based on RLink, a replacement (or a layer above) rJava.

Windows

Many mentions of **rcom** and **statconnDOM** (R–C# link) to interface R and Excel (there is even a book on the

subject – frightening) or generate powerpoint files.

R Ecosystem

In many domains, R provides very similar functions in different packages each with its own syntax. Some (well needed but) isolated **unification** works have begun, for instance **Zelig** (statistical models), **partykit** (recursive partitioning algorithms) or **optimx** (for optimization: it provides a single interface and compares the various algorithms, to help the user choose the best for the problem at hand).

Instructional presentations included **sparse matrices** with the **Matrix** package (used for linear mixed models by **lme4**) and spatial autocorrelated (SAR) models; **S4 classes** (the dispatch is based on the type of all arguments, not just the first as with S3 classes); implementing new statistical distributions (the **dpqr** functions, and a few others), as exemplified in the **distr** and **VarianceGamma** packages, with some good general **software engineering** advice.

Provenance tracking tries to answer the question “where does this variable come from? how was it modified?”; it is implemented in CXXR (a C++ reimplementation of R, designed to easily produce experimental versions of the R interpreter): the **pedigree** function is similar to **history**, but the *audit trail* it returns only gives the commands that affected a given object. The Open Provenance model aims at provenance tracking across systems.

Reproducible computing can be easily obtained by recording (“blogging”) everything you type, including outputs and plots, in a way reminiscent of Maple worksheets. Social data networks (Many Eyes, Swivel, the-data.org, etc.) are starting to appear (but they often present pure coincidences as hard evidence).

Multivariate copulas at work

M. Fischer

RMetrics Workshop 2009

While binomial copulas are galore, “higher dimensional” ones (dimension 3 or 4, not the 200 or 2000 I needed a few years ago) used to be rare. The situation is slowly changing:

- Elliptical copulas;
- Archimedian copulas

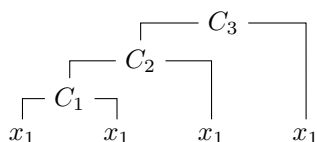
$$C(x_1, \dots, x_n) = \phi^{-1}(\phi(x_1) + \dots + \phi(x_n))$$

which can be written multiplicatively (set $\theta(x) = \exp -\phi(x)$)

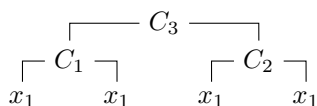
$$C(x_1, \dots, x_n) = \theta^{-1}\left(\prod_i \theta(x_i)\right)$$

- **Generalized archimedian copulas**: use archimedian copulas to group variables two at a time, in a

fully-nested way



or using an arbitrary tree;



- **Generalized multiplicative archimedian copulas:** replace the independent copula \prod by another copula C_0

$$C(x_1, \dots, x_n) = \theta^{-1}(C_0(x_1, \dots, x_n));$$

this can be seen as a $(\mathfrak{S}_n$ -symmetric) deformation of C_0 ;

- **Liebscher copulas** combine several multiplicative archimedian copulas

$$C(x_1, \dots, x_n) = \Psi\left(\frac{1}{m} \sum_j \prod_i \theta_j(u_i)\right);$$

- The **Koehler-Symanowski copulas** are build as follows: take n iid random variables

$$Y_1, \dots, Y_n \sim \text{Exp}(\lambda = 1);$$

choose random variables to model bivariate associations

$$G_{ij} \sim \Gamma(\alpha_{ij}, 1), \quad i < j;$$

(and trivariate associations if you want: $G_{ijk} \sim \Gamma(\alpha_{ijk}, 1)$) and try to put them all in a formula, e.g.,

$$U_i = \left(1 + \frac{Y_i}{\sum_j G_{ij}}\right)^{\sum_j \alpha_{ij}}$$

- Starting from a (graphical model) decomposition

$$f(x_1, x_2, x_3) = f(x_3)f(x_2|x_3)f(x_1|x_2, x_3)$$

one can express a copula as a (complicated) product of bivariate copulas; this is the **pair copula decomposition** (PCD).

A maximum likelihood fit of those on several financial datasets suggests to forget plain archimedian copulas: elliptic (Student) or pair copulas provide a better fit.

Pair-copula constructions of multiple independence

K. Aas et al.

Detailed presentation of pair copulae. The decompositions are not described by a single graphical model but by a (maximal) nested set of (tree) graphical models – a D -vine. Imposing some conditional independence greatly simplifies the model.

Package development in R: Implementing GO-GARCH models

B. Pfaff

RMetrics Workshop 2009

The univariate GARCH model can be extended to a multivariate model by direct generalization (VEC, BEKK, factor models: keep the same formulas, with matrices instead of real numbers, and impose some conditions on them to keep the number of parameters reasonable); by linear combinations of univariate GARCH models (GO-GARCH (generalized orthogonal), latent factor GARCH) or even non-linear combinations (dynamic conditional correlation, general dynamic covariance models).

Consistent return and risk forecasting for portfolio optimization using kernel regressions

J. Hindebrand and T. Poddig

RMetrics Workshop 2009

Separately estimating expected returns and risk model (variance matrix) is suboptimal (when your only source of information is the time series of returns). The authors suggest to estimate both from past returns using *kernel regression* (apparently sometimes called “general regression neural network” (GRNN)) which can be seen as a smoothed k nearest neighbours algorithm. The risk can be estimated “explicitly”, from the residuals

$$\hat{\sigma}^2 = \frac{1}{k} \sum (\hat{y}_i - y_i)^2$$

or “implicitly”

$$\hat{y} = \frac{1}{k} \sum y_i$$

$$\hat{\sigma}_i = \frac{1}{k} \sum (\hat{y} - y_i)^2$$

(the sums are over the k nearest neighbours; for kernel regression, replace those averages by weighted averages).

Financial crises and the exchange rate volatility for Asian economies

T. Oga and W. Polasek

RMetrics Workshop 2009

Markov switching AR-GARCH models are amenable to bayesian MCMC estimation.

Markowitz and two managers

K. Rheinberger

RMetrics Workshop 2009

In many companies, portfolio managers work independently and their strategies or portfolios are combined by higher authorities (*i.e.*, they decide on the capital allocated to each). This is suboptimal: if the asset classes are correlated, the optimal global portfolio need not be obtainable as a combination of efficient subportfolios.

Simple parallel computing in R with Hadoop
RMetrics Workshop 2009

MapReduce (also called **Cloud Computing** or **SAAS** (software as a service)) is a trendy name for “distributed divide and conquer” where the task division step is aware of the location of the data on the *distributed file system* (DFS) to limit network usage. R can use *Hadoop*, a (Java) MapReduce implementation, initially developed by Yahoo and now maintained by the Apache foundation.

**Modeling and evaluation
of insurance risk using actuar**
RMetrics Workshop 2009

The **actuar** package provides some functionalities for actuarial science, which is just “risk management” with a different vocabulary:

- Distributions often used to model losses (on top of the usual dpqr functions, you also have moments $E[X^k]$, limited moments $E[\text{Min}(X, x)^k]$ and moment generating functions $E[e^{tX}]$);
- *Minimum distance estimators* (MDE) for the Cramer-von Mises, modified χ^2 or layer average severity distances (these are not clearly defined);
- Handling (plotting, fitting) of censored data (both sides);
- Various estimators of the distribution of $X_1 + \dots + X_N$ from those of N and the X_i (assumed iid): Panjer algorithm (uses discretized distributions), simulations, normal approximations (these algorithms are not detailed);
- Ruin models, *i.e.*, estimation of

$$P[\exists t > 0 : U(t) < 0]$$

where

$$\begin{aligned} U(t) &: \text{surplus at time } t \\ u &: \text{initial surplus} \\ c_k &: \text{premium collected at time } k \\ N(t) &: \text{number of claims in } [0, t] \\ X_n &: \text{value of the } n\text{th claim} \\ U(t) &= u + \sum_{k \leq t} c_k - \sum_{n \leq N(t)} X_n \end{aligned}$$

- from the distributions of N and X ;
- Simulations of hierarchical models.

An overview of random number generation
C. Dutang
RMetrics Workshop 2009

The default random number generator (RNG) in R (the **runif** function) is the **Mersenne twister** (MT), a linear congruential RNG modified by some bitwise operations. The **randomtoolbox** package implements other RNG (mainly generalized MT: WELL, SFMT) and quasi-random number generators (aka **low discrepancy sequences**):

- For the **Van der Corput** sequence, choose a prime number p , write successive numbers $1, 2, 3, \dots, n$ in base p , $n = \sum_j a_j p^j$ and return $\phi_p(n) = \sum_j a_j / p^{j+1}$; if you choose several prime numbers, you have a multidimensional low discrepancy sequence, the **Halton sequence**; in R, the **halton(n,k)** function returns the first n elements of the k -dimensional Halton sequence;
- The **torus algorithm** (or **Kronecker sequence**) is

$$u_n = (\text{frac } n\sqrt{p_1}, \dots, \text{frac } n\sqrt{p_d})$$

where $\text{frac } x = x - \lfloor x \rfloor$ is the fractional part.

RNGs can be tested by looking at

- The histogram of generated data;
- The autocorrelation function (ACF);
- The order test, which checks in which order x_{i-1} , x_i and x_{i+1} are);
- The Poker test: take d consecutive numbers from your sequence, count how many there are in each $[(k-1)/d, k/d[$ interval ($k \in \llbracket 1, d \rrbracket$), compare with the expected values using a χ^2 test.

Discrepancy can be defined as

$$\begin{aligned} D_n(x, A) &= \frac{1}{n} \# \{ i \in \llbracket 1, n \rrbracket : i \in A \} \\ D_n^\infty(x, \mathcal{P}) &= \sum_{A \in \mathcal{P}} |D_n(x, A)| \end{aligned}$$

for some $\mathcal{P} \subset \mathcal{P}([0, 1]^d)$.

GMM and GEL with R
P. Chaussé
RMetrics Workshop 2009

The *generalized method of moments* (GMM) fits a model to your data by considering several quantities or *moments* (e.g., average, variance, median, quantiles, higher moments, truncated moments, etc.) and fine-tuning the parameters so that the theoretical quantities be as close from the observed ones as possible. More precisely, find several moments $g = (g_1, \dots, g_n)$ so that $E[g(\theta, X)] = 0$ and compute

$$\begin{aligned} \hat{\theta} &= \underset{\theta}{\text{Argmin}} \|\bar{g}(\theta)\|^2 \\ \bar{g}(\theta) &= \frac{1}{T} \sum_t g(\theta, x_t). \end{aligned}$$

Since the moments are not “independent”, the Euclidean distance is not the best choice: you will prefer

$$\hat{\theta} = \underset{\theta}{\text{Argmin}} \bar{g}(\theta)' W \bar{g}(\theta)$$

for some W . The most efficient estimator is obtained with

$$W = \sum_{s \in \mathbf{Z}} \text{Cov}(g(\theta, x_t), g(\theta, x_{t+s}))$$

which can be estimated (with an HAC (heteroscedasticity and autocorrelation consistent) estimator) as

$$\hat{\Omega}(\theta^*) = \sum_{|s| \leq T-1} k_h(s) \hat{\Gamma}_s(\theta^*)$$

$$\hat{\Gamma}_s(\theta^*) = \frac{1}{T} g(\theta^*, x_t) g(\theta^*, x_{t+s})'$$

k_h : some kernel

For instance, one could try any of the following:

- Estimate θ^* with $W = 1$; compute $\hat{\Omega}(\theta^*)$; estimate $\hat{\theta}$;
- Iterate the above until convergence;
- Optimize properly:

$$\hat{\theta} = \underset{\theta}{\text{Argmin}} \bar{g}(\theta)' \hat{\Omega}(\theta) \bar{g}(\theta).$$

The limiting distribution is known.

In finance or economy, you can use the CAPM, the utility function and other meaningful quantities to define moments.

In R, just provide your moments and your data to the `gmm` function.

I did not understand what GEL (generalized empirical likelihood) was.

***A tale of two theories:
Reconciling random matrix theory
and shrinkage estimation
as methods for covariance matrix estimation***
B. Rowe
RMetrics Workshop 2009

Both methods try to remove the noise in the spectrum of the sample covariance matrix: random matrix theory (RMT) violently shrinks the low values to zero; shrinkage smoothly shrinks all values to their mean (or to a predefined value). Combining the two methods turns out to be a bad idea.

***The structure and function
of complex networks***
M.E.J. Newman
arXiv:cond-mat/0303516

Given a network, one can compute the following numeric quantities (invariants):

- Harmonic mean geodesic distance between all pairs (the arithmetic mean would have problems with non-connected vertices);
- Clustering coefficient:

$$C = \frac{1}{n} \sum C_i$$

$$C_i = \frac{\text{Number of triangles connected to vertex } i}{\text{Number of triples centered on vertex } i};$$

- Degree distribution;
- Network resilience: progressively remove vertices, starting with the highest degree ones, and plot of the mean distance between edges versus the fraction of vertices removed;

- **Assortativity** coefficient, to measure **mixing patterns** between vertex types

$$r = \frac{\text{tr } e - \|e^2\|}{1 - \|e^2\|}$$

where e_{ij} is the fraction of edges between vertices of types i and j ; those types can be numeric (e.g., the degree, leading to the **degree correlation**) and even continuous.

Here are a few models of random graphs:

- *Poisson random graphs* (choose the degree of each vertex; join the vertices at random), aka *Erdős–Rényi graphs*;
- Configuration model (replace the Poisson distribution by another one); they fail to account for **transitivity**;
- Exponential random graphs: choose a few numeric properties of graphs, $\varepsilon_1, \dots, \varepsilon_k$, e.g., number of vertices, number of vertices of a given degree, etc.; choose a few numbers β_1, \dots, β_k (inverse temperatures); sample graphs with probability

$$P(G) \propto \exp - \sum_i \beta_i \varepsilon_i(G).$$

These are amenable to (Monte Carlo) simulations, but not analytical derivations, except in the case of **Markov graphs** (the absence/presence of edges are independent unless the edges share a vertex). Unfortunately, they model transitivity in an unnatural way: there are too many complete cliques.

- *Small world model*: put all the vertices on a circle (or any other low-dimensional lattice), connect each vertex to its k nearest neighbours, rewire edges at random with probability p (a variant adds edges with probability p : this can be seen as the average of a lattice and a random graph);
- *The Barabási-Albert model* is a model of network growth, *i.e.*, it tries to explain where the graph properties come from: add vertices one by one, link them to previous nodes, with a higher probability for nodes with a high degree (preferential attachment); the earlier Price model was similar but used directed edges.

Fully mixed epidemiological models, such as SIR (susceptible-infected-recovered: after catching the disease, people (die or) become immune)

$$\dot{S} = -\beta IS$$

$$\dot{I} = \beta IS - \gamma I$$

$$\dot{R} = \gamma I$$

or SIS (reinfections are possible)

$$\dot{S} = -\beta IS + \gamma I$$

$$\dot{I} = \beta IS - \gamma I$$

can be generalized to a network. Other processes on networks include **percolation** and search.

***strucchange: an R package
for testing for structural change
in linear regression models***
A. Zeileis et al.

The **strucchange** package implements tests based on the following **empirical fluctuation processes**:

- OLS-CUSUM: cumulated sum of the residuals (a brownian bridge);
- recursive CUSUM: cumulated sums of the residuals, where the i th residual comes from the model estimated on the first i observations;
- Recursive MOSUM: idem, on a moving window;
- Fluctuations: parameters of the model estimated on an expanding window, suitably renormalized (contrary to the residuals, this is a multidimensional process);
- Moving estimate: idem, on a moving window.

In case of a structural change, these processes “fluctuate more”; they can be compared with boundaries of the form $b(t) = \lambda$ (MOSUM: it is stationary), $b(t) = \lambda\sqrt{t}$ (brownian motion) or $b(t) = \lambda\sqrt{t(1-t)}$. However, the crossing probability is easier to compute for linear boundaries: we lose some power at the beginning and end of the interval.

```
r <- epf(y ~ x, type="fluctuation")
plot(r)
sctest(r)
```

Instead of those *significance tests*, which have no clear alternative hypothesis, you may prefer an F test, which tests against the alternative that there are two different models, one before and one after an (unknown) breakpoint; they compare the sum of squared residuals for the model with and without a breakpoint at observation τ with an F statistic F_τ and aggregate them as $\sup F_\tau$, $\text{Mean } F_\tau$ or $\log \text{Mean exp } F_\tau$.

```
f <- Fstats(y~x)
sctest(f)
```

The package also provides **monitoring**.

***A unified approach
to structural change tests
based on F statistics,
OLS residuals and ML scores***
A. Zeileis

The following classes of structural change tests are actually generalized M -fluctuation tests, built from the empirical fluctuation process (efp):

- CUSUM (the absolute maximum of the cumulative sums of the residuals rescaled by an estimate $\hat{\sigma}^2$ of the error variance – it is not a real test, there is no clear alternative hypothesis, but it is often used as an exploratory tool);
- SupLM = $\sup_{t \in [0,1]} \frac{\|\text{efp}_t\|_2^2}{t(1-t)}$ (in the alternative, the

model parameters have two values, before and after some (unknown) breakpoint);

- The Nyblom-Hansen statistic, where in the alternative H_1 , the parameters follow a random walk, boils down to

$$\frac{1}{n} \sum_{i=1}^n \left\| \text{efp}_{i/n} \right\|_2^2.$$

To detect multiple breakpoints, you may want to use a moving sum (MOSUM) instead of a cumulative sum.

These tests can be extended to a **monitoring** setup, where the observations arrive one by one: estimate the model on $[0, 1]$ and compute the empirical fluctuation process beyond 1; the critical values become a curve $b(t)_{t>1}$ instead of a constant.

Permutation tests for structural change
A. Zeileis and T. Hothorn

The SupF (or SupLM: the test statistic can be transformed to look like an F or a Student T one) tests whether the average of a series of random variables $(X_{k/n})_{k \in \llbracket 1, n \rrbracket}$ is a constant μ against the alternative hypothesis H_π that it is μ_1 before some breakpoint $\tau \in [0, 1]$ and μ_2 after, by considering the statistic

$$\frac{\text{RSS}_\pi}{\text{RSS}_0},$$

suitably renormalized

$$Z_\pi = \sqrt{(n-1) \left(1 - \frac{\text{RSS}_\pi}{\text{RSS}_0} \right)},$$

where RSS_0 and RSS_π are the sums of squared residuals under H_0 and H_π , and aggregating them as

$$D = \text{Max}_\pi Z_\pi.$$

This statistic can be compared with the following distributions:

- \mathcal{D}_∞ : the *unconditional limiting distribution*, i.e., the limiting distribution, for $n \rightarrow \infty$; it is

$$\sup_{\pi \in [0,1]} \pi(1-\pi)B(\pi),$$

where B is a brownian bridge;

- $\mathcal{D}_{\sigma|Y}$: the *conditional distribution*, obtained by randomly permuting the observations (usually computed via simulations; only in very rare cases is it analytically tractable);
- $\mathcal{D}_{\infty|Y}$, the *asymptotical conditional distribution*, which can be computed analytically:

$$Z \sim N(0, \Sigma)$$

$$Z_{\pi, \tau} = \frac{\pi(1-\tau)}{\sqrt{\pi(1-\pi)\tau(1-\tau)}} \quad (\pi \leq \tau)$$

This can be generalized to structural changes in binary variables, multivariate series, stratified observations, parametric models.

Implementing a class of structural change tests: an econometric computing approach
A. Zeileis

Structural change tests can be performed as follows:

- Compute the residuals of your model, $(r_i)_{i \in \llbracket 1, n \rrbracket}$;
- Robustify those residuals, $(\psi(r_i))_{i \in \llbracket 1, n \rrbracket}$, where ψ is an M -estimation scale function; we only want the sum of the robustified residuals to be zero;
- Compute their cumulative sum process $(Z_t)_{t \in [0, 1]}$ (the article is not very clear if we take the cumulative sum of the residuals or build a process from the model coefficients estimated on an expanding window: we cannot get a multidimensional process with just the residuals);
- Estimate the asymptotic covariance matrix of the scores \hat{J} ; we then have

$$\text{Cov}(Z_t, Z_s) = \text{Min}(t, s)\hat{J};$$

you can (should) use an HAC (heteroskedasticity and autocorrelation consistent) estimator for J ;

- Compute the decorrelated fluctuation process

$$\text{efp}_t = \hat{J}^{-1/2} Z_t;$$

it converges to a brownian bridge (since the residuals sum up to zero, we have $Z_0 = Z_1 = 0$); you may choose to use the whole matrix \hat{J} or just its diagonal elements;

- Aggregate this multidimensional process, for instance:
 - Take the maximum of the components at each point in time, then take the maximum accross time (maxBB);
 - Take the L^2 norm of the components at each point in time; then take the average over time (Cramer von Mises statistic);
 - Take the range (*i.e.*, Max – Min) over time, for each component; then take the maximum of those ranges (range test).

This is implemented in the **strucchange** package and will work for any model for which you can compute the residuals (Poisson, logistic, beta regressions, etc.).

Minority games
C.H. Yeung and Y.-C. Zhang
arXiv:0811.1479

A readable review article.

Copula-based non-linear quantile autoregression
X. Chen et al.

A copula-based Markov model (for a stationary time series $(Y_t)_t$) just specifies the copula C of (Y_{t-1}, Y_t) and the marginal distribution of Y_t . This is often estimated by a 2-step procedure: first, estimate the marginal distribution, then, estimate the copula (using either the empirical marginal distribution or the fitted one).

One can look at the τ th quantile of $Y_t | Y_{t-1} = x$:

$$F^{-1}(C(\tau, \cdot)^{-1}(F(x))).$$

The article considers the case of a parametric copula and marginal distribution, but those parameters depend on the quantile τ . This is more robust to model misspecification.

Quantile autoregression
R. Koenker and Z. Xiao (2006)

Quantile autoregression (QAR) is a special case of **random coefficient autoregression** (RCAR)

$$y_t = \theta_0(U_t) + \theta_1(U_t)y_{t-1} + \dots + \theta_p(U_t)y_{t-p}$$

$$U_t \sim U(0, 1).$$

For instance

$$\theta_0(\tau) = \sigma \Phi^{-1}(\tau)$$

$$\theta_1(\tau) = \text{Min}(\gamma_0 + \gamma_1 \tau, 1)$$

leads to a mixture of unit root or explosive behaviour (when $\theta_1 = 1$) and mean-reverting behaviour (when $\theta_1 \in (0, 1)$) – this is actually sufficient to make the process stationary.

Since the functions θ are unknown, this is a non-parametric model. (Most of the theoretical troubles come from the requirement that those functions be monotonic.) You can test for *asymmetric dynamics* by checking if θ_1 depends on τ at all.

March madness, quantile regression bracketology and the Hayek hypothesis
R. Koenker and G.W. Bassett

More details on this example.

Issues on quantile autoregression
J. Fan and Y. Fan

Quantile autoregression (QAR) is related to **functional coefficient autoregression** (FCAR) models

$$y_t = \alpha_0(U_t) + \alpha_1(U_t)Y_{t-1} + \dots + \alpha_p(U_t)Y_{t-p} + \varepsilon_t$$

where U_t is observed. The special case $U_t = Y_{t-d}$ is known as **functional autoregression** (FAR). Identifiability is a real problem, and in case of lack of monotonicity, the estimators are not consistent.

Feature selection in “omics” prediction problems using cat scores and false non-discovery rate control
M. Ahdesmäki and K. Strimmer
arXiv:0903.2003

Linear discriminant analysis (LDA) is a supervised learning algorithm that assumes the data is a mixture of gaussian distributions with the same variance matrix Σ ; the LDA discriminant score of a new observation x for cluster k is

$$d_k(x) = \log P[k|x] = \mu'_k \Sigma^{-1} x - \frac{1}{2} \mu'_k \Sigma^{-1} \mu_k + \log \pi_k$$

where μ_k is the center of cluster k and π_k its weight (or a priori probability). Estimating the inverse Σ^{-1} is problematic: **diagonal discriminant analysis** (DDA, aka naive Bayes classification) assumes it is diagonal. The nearest shrunken centroids (NSC, aka PAM, partitioning around medoids) algorithm is a regularized version of it.

The article suggests a regularized LDA (regularize the variances (shrink to the median) and the correlation matrix (ridge regularization) separately) to account for correlations, *i.e.*, for non-spherical clusters.

Monitoring networked applications with incremental quantile estimation

J.M. Chambers et al.
arXiv:0708.0302

Sequential quantile estimation algorithms compute quantiles in a “cumulative” (online) way, as data arrives:

- Exact algorithm exist (e.g., based on the quick sort) to track a single quantile, but they have to keep a potentially large buffer of data; they can be modified to track a set of quantiles;
- Stochastic approximation is similar, but assumes the data has a continuous distribution to approximate between the quantiles.

Incremental quantile (IQ) estimation is a quick-and-dirty algorithm that allows both online quantile estimation and *aggregation* of quantiles from distributed agents; it uses cumulative distribution functions (CDF), which are easy to aggregate, rather than quantiles:

- On each agent, wait until you have enough data; turn it into a cumulative distribution function (CDF); blend it with the previous CDF estimate;
- From time to time, send it to the server, and start the computation afresh;
- On the server, aggregate the CDFs received from the agents and compute the desired quantiles.

The CDFs can be stored and sent as a set of quantiles (for increased precision, we may want more quantiles than the final users actually receive).

This can be useful in distributed QoS (quality of service) measurement.

Text mining infrastructure in R

I. Feinerer et al.
Journal of statistical software (2008)

The **tm** package provides a framework to manage collections of texts (with their metadata; to avoid memory problems, you can just keep the metadata in memory and leave the texts on disk) and relies on other packages for stemming (**Rstem**, **snowball** (Weka)), synonym substitution (**wordnet**), part-of-speech tagging (**openNLP**), term-document matrix computation (**Matrix**, for sparse matrices), machine learning algorithms (for unsupervised learning (document clus-

tering): hierarchical clustering, k -means; supervised learning (document classification): k -nearest neighbours (kNN), support vector machines (SVM); or more text-specific tasks: information retrieval, keyword extraction), string kernels (**kernlab**).

The **string kernel** of two strings x and y is the scalar product $[\Phi(x), \Phi(y)]$ of their embeddings in some high-dimensional space: knowing this scalar product is sufficient for most linear-algebra-based algorithms, the actual coordinates are not needed. Examples include:

- $k(x, y) = \sum_{s \sqsubseteq x, y} \mathbf{1}_{s_0=s_{|s|}} =$ (bag of words)
- $k(x, y) = \sum_{s \sqsubseteq x, y} \mathbf{1}_{|s| \leq n}$ (full string kernel)
- $k(x, y) = \sum_{s \sqsubseteq x, y} \mathbf{1}_{|s|=n}$ (string kernel)
- $k(x, y) = \sum_{s \sqsubseteq x, y} \lambda^{|s|}$ (substring kernel).

Forecasting the term structure of government bond yields

F.X. Diebold and C. Li (2003)

The forward rate can be approximated as

$$f_t(\tau) = \alpha_t + \beta_t e^{-\lambda_t \tau} + \gamma_t \lambda_t e^{-\lambda_t \tau}$$

where the three time series α (level), β (slope) and γ (curvature) can be estimated with an AR model (why did λ disappear?).

Comparing smooth transition and Markov switching autoregressive models of US unemployment

P.J. Deschamps (2007)

Comparison of a Markov switching autoregressive model (MSAR) and a logistic smooth transition autoregressive model (LSTAR)

$$y_{t+1} = (\alpha_0 + \alpha_1 y_t) + G_t(\beta_0 + \beta_1 y_t) + \varepsilon_t$$

$$G_t = \frac{1}{1 + e^{-(a s_t + b)}}$$

where s_t is *observed*, e.g., $s_t = y_t - y_{t-12}$. There are variants with other shapes for the transition function G (it need not be symmetric) and/or more than two states.

Two-dimensional correlation optimized warping algorithm for aligning GC×GC-MS data

D. Zhang et al.

Two-dimensional time warping can be applied to spectrometry.

Fast text mining using kernels in R

I. Feinerer and A. Karatzoglou

String kernels

$$k(x, x') = \sum_{s \in A^*} \text{num}_s(x) \text{num}_s(y) \lambda_s$$

usually require $O((|x| + |y|)^2)$ operations, but they can be computed from the **suffix tree** of the texts in linear time; using a **suffix array** (not defined), as implemented in the **kernlab** package is (still linear but) more efficient.

Spectral clustering is just a complicated name for clustering (using traditional techniques) after PCA-dimension reduction using the kernel trick.

Invariant coordinate selection

D.E. Tyler et al.

Most multivariate data analysis methods are based on the variance matrix, or more generally *one* measure of dispersion. **Invariant coordinate selection** (ICS) uses *two* measures of dispersion, V_1 and V_2 , and looks at the eigenvalues and eigenvectors of one wrt the other, *i.e.*, looks for $p \in \mathbf{R}$ and $h \neq 0$ such that $V_2 h = \rho V_1 h$; these are also the eigenvalues of $V_1^{-1/2} V_2 V_1^{-1/2}$. For elliptical data, any two measures of dispersion are proportional $V_2 = \lambda V_1$: ICS highlights departure from gaussianity.

Examples of scatter (or dispersion) matrices include:

- The variance matrix;
- The weighted variance (the weights could depend on the Mahalanobis distance);
- M -estimates of the variance, *i.e.*, an adaptively-weighted variance;
- Higher breakdown robust scatter matrices: MVE, MCD, S -estimates, τ -estimates, etc. (they are computationally expensive);
- A dispersion matrix wrt to the origin (*i.e.*, the location estimator is constant);
- A dispersion matrix of the pairwise differences, $\text{Disp}(x_i - x_j)_{i,j}$
- Fourth central moments;
- Tyler's shape matrix.

Outliers are often spotted by looking at the Mahalanobis distance for one measure of dispersion: besides (before) a plot of the first eigencoordinates, a scatterplot of those distances for several measures of dispersion may also be helpful.

Tools for exploring multivariate data: the package ICS

K. Nordhausen et al.

Journal of Statistical Software (2008)

A more application-oriented presentation of invariant coordinate selection (ICS). The ICS transformation can be described as:

- Whiten the data wrt the first measure of dispersion: $x \mapsto (x - 1'\bar{x})\text{Disp}^{-1/2}$ (stopping after that step is not satisfactory: the square root is not unique and the whitening transformation is not affine-equivariant);
- Perform a principal component analysis (PCA) on the whitened data, wrt the second measure of dispersion.

Scala by Example

M. Odersky

Scala is a functional and object-oriented language, with static typing and type inference (somewhat inferior to Haskell), whose main advantages are:

- It is a script-like language, easy to write and read;
- It runs on the **Java Virtual Machine** (JVM) and can therefore leverage all existing Java libraries and the Java infrastructure and knowledge you may already have;
- It provides a rather exhaustive implementation of parallelism patterns: Erlang's **actors**, lazy variables, explicit parallelism, Java threads, etc.;
- Method names can contain weird characters (*:, etc.), facilitating the creation of libraries that look like domain-specific languages (DSL).

Its main drawbacks are:

- The type inference is not as good as Haskell's (e.g., you have to provide the return type of recursive functions);
- Function types can be unreadable, with the name of the variable (function) in the middle of the type, as in C;
- There are still too many parentheses.

Here is a summary of the syntax.

When you define a variable, it can be left unevaluated (**def**, this is useful for functions, which are just blocks), immutable (**val**), mutable (**var**), lazy (**lazy val**) or evaluated in the background (**var x = future(f(1))**).

```
def f(
  g : Int => Int, // Function argument
  a : Int,
  b : => Int, // Not evaluated immediately
): Int = { ... }
class Foo(a:Int, b:Int) {...} // private members
class Bar(...) Extends Foo {
  ...
  override def toString ...
}
val x = new Foo(1,2);
```

A **trait** is a Java interface, designed to be added to a class, *i.e.*, designed for multiple inheritance; abstract classes are designed for single inheritance.

Case classes provide implicit constructors, implicit getter methods and pattern matching, *i.e.*, implicit tests on the type of objects – this is just syntactic sugar, but it increases code readability.

```
abstract class A
case class A1(n:Int) extends A
case class A2(n:Int, m:Int) extends A
...
e match { case A1(n) => n
          case A2(n,m) => n+m }
```

Type parameters (aka templates or generic types)

can also specify lower and upper bounds on types.

```
class Set[A]
// A implements the Ordered trait:
class Set[A <: Ordered[A]]
// A can be converted into
// something that implements Ordered
class Set[A <% Ordered[A]]
```

Inheritance relations can be automatically **lifted** to generic types, so that if $A <: B$, then $\text{Stack}[A] <: \text{Stack}[B]$. In some cases, this mapping is not covariant but contravariant: e.g., if $A <: B$, then $\text{Fun}[B, C] <: \text{Fun}[A, C]$.

```
class Stack[+A]
class Fun[-A, +B]
```

Functions are just objects with an **apply** method; $f(x)$ is just a shorthand for $f.\text{apply}(x)$; arrays implement the same interface and use the same syntax.

Tuples, written (a, b) , are just a shorthand for the $\text{Tuple2}[A, B]$ type.

Besides **Array** (Java's mutable arrays), Scala provides a **List** type (homogeneous, immutable, unsuitable for random access), as in Lisp, with the usual **map**, **foreach**, **forall**, **exists** (called **all** and **any** in some other languages) methods. The **for** comprehension is a shorthand for those list operations.

```
for(x <- xs if f(x)) yield g(x)
```

Streams are lazy (potentially infinite) lists; they have (almost) the same methods. Iterators, *i.e.*, classes with **hasNext** and **next** methods, are an imperative analogue of streams; prefer them to streams if you know previous values can be discarded.

Methods can have implicit arguments (actually, even the method name can be implicit: these are implicit conversions (casts)), but this does not look as useful the named and implicit arguments of most scripting languages.

Detailed examples include an auction service (actors), type inference, electric circuit simulation, N -queen problem (list comprehensions), building **N**, Newton's method and more general fixed point problems, stacks (type parameters).

This introduction does not mention some important features of Scala, such as the ability to include XML directly in the code (not as strings), with the $\langle \rangle$ delimiters, in the same way most languages allow you to enter strings directly (not as arrays of characters), with the `"` delimiter.

Regularized discriminant analysis
J.K. Friedman
Journal of the American Statistical
Association (1989)

Your data comes from a mixture of distributions, $\sum_k \lambda_k f_k(x)$, and you want to find from which class k a given observation x comes from. If you knew the prior probabilities λ_k and the conditional distributions f_k , you would just use the Bayes rule. If those conditional distributions are multivariate gaussian with identical (resp. different) variance matrices, obtained from the plug-in (aka sample) estimators, this is linear discriminant analysis (LDA) (resp. quadratic discriminant analysis (QDA)). Some people also use diagonal discriminant analysis (DDA).

You can use a regularized estimator of those variance matrices.

Partial least squares: a versatile tool for the analysis of high-dimensional genomic data
A.-L. Boulesteix and K. Strimmer (2006)

Should you want formulas to compute partial least squares (PLS) yourself, here they are. Let X be the predictive variables (one observation per row, few rows, many columns) and Y the variable to predict. The data is assumed to come from latent components T ,

$$X = TP' + \text{noise}$$

$$Y = TQ' + \text{noise}$$

Once you have those latent components T , you can estimate \hat{Y} by least squares. The latent components are of the form $T = XW$, where W is defined by some iterative optimization procedure (there are many of them: PLS1, PLS2, NIPALS, Kernel-PLS, SIMPLS, etc.) such as

$$\begin{aligned} \forall i \quad & \text{Maximize } w_i' X' Y Y' X w_i \\ & \text{such that } w_i' w_i = 1 \\ & \text{and } \forall j < i \quad w_i' X' X w_j = 0. \end{aligned}$$

In R, check the **pls**, **pls.pcr**, **plsgenomics**, **gpls** packages.

Multi-objective optimization using genetic algorithms: a tutorial
A. Konak et al.
Reliability engineering and system safety
(2006)

Multiobjective optimization do not lead to a *single* but to a *set* of solutions (the Pareto set, aka efficient frontier) – transforming the problem into a single-objective one is rarely satisfactory: it involves arbitrary choices (weights, utility function) that often have a huge (and badly understood) effect on the solution.

Genetic algorithms (GA) naturally produce a population of solutions, but care should be taken: the solutions found should

- be uniformly distributed (genetic algorithms tend to produce clusters (genetic drift): to prevent this, you can try to decrease the fitness of densely populated areas)

- explore the whole spectrum of the efficient frontier (genetic algorithms tend to focus on the middle and miss the extremities – you may want to store non-dominated solutions in a separate list, to avoid losing them).

There is a very large number of multi-objective genetic algorithms: the authors provide a comparison table (check PAES, NSGA-II, SPEA-2).

Multi-objective generic algorithms: problem difficulties and construction of test problems

K. Deb

A set of test problems for multi-objective genetic algorithms.

Spectral graph theory and its applications

D.A. Spielman (2004)

An interesting applied linear algebra course, that explains how to read the properties of a graph from the eigenvalues and vectors of its incidence matrix or Laplacian matrix. The Laplacian of an edge is $\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$ and the Laplacian of a graph is the sum of those of its edges. Applications include graph colouring, error correcting codes, random matrices and random graphs, graph approximation (find a sparser graph whose Laplacian matrix is close, in spectral norm – this can be used to speed up linear system solving).

Maximum overhang

M. Paterson et al.

arXiv:0909.0093, arXiv:0910.2357

Funny...

An evaluation of mathematics competitions using item response theory

J. Gleason

Notices of the AMS (2008)

An introduction to item response theory (IRT).

Little B: Biological modeling

Little B is a Lisp-like biological simulation language – basically, to define large systems of ordinary differential equations (ODE) from simple blocks. Only have a look if you think Lisp has too few parentheses...

OpenModelica Users Guide

P. Fritzson et al.

Modelica is a Pascal-like, object-oriented, equation-based, domain-specific language (DSL), targeted at physical (event-based) simulations, probably similar to Matlab' Simulink; there are interfaces to/from Matlab or Maple. OpenModelica is a free implementation with an Eclipse-based IDE (integrated development environment).

WaveScript User Manual

A (functional) language to describe (and run) data flow graphs. The compiler is written in Scheme and converts the code to C or MLtron (there is also an unsupported C++ backend through XStream).

It is not a “higher order data flow language”: the graph does not change at runtime.

Scale theory, serial theory and voice leading

D. Tymoczko (2007)

When turning a harmonic progression into a set of melodies (*i.e.*, you have a chord voicing, often spread over several octaves, you have chosen the next chord, but you still have to decide on the voicing), composers try to use *efficient voice leading*, *i.e.*, try to limit how far each voice moves; this facilitates the perception of independent musical lines (in addition, you may want those voice leadings to avoid crossings and be bijective). The number of voice leadings may look daunting (especially for large chords, *e.g.*, hexachords), but the problem is not.

Freer than Max – porting FTM to Pure Data

I. m Zmölzig (2007)

The FTM library, that provides complex data structures to data flow languages, is now available for Pure Data – it is an alternative to Pure Data's “graphical data structures” and the PDCContainer (C++ STL) library.

Analytical features

to extract harmonic or rhythmic information

G. Cabral et al.

Description of an automated musical feature (note, chord, rythm, timbre) extraction tool; it just combines known algorithms:

- Time-based pitch recognition algorithms compare (by looking at the correlation or just the distance) the signal with its lags; it works better for low frequency signals;
- Frequency-based pitch recognition algorithms (*e.g.*, the harmonic product spectrum (HPS)) try to find a frequency f for which the spectrum has peaks at $2f$, $3f$, $4f$, etc. (I expect this to fail for lamellophones); the cepstrum is mentionned but not defined;
- For better pitch recognition, you can combine several algorithms;
- Chord recognition algorithms split the spectrum into regions, compute the energy in each, assign it to the 12 notes, normalize the resulting vectorm and apply some machine-learning algorithm.

Can the Red Queen help catch the Snark?

A co-evolutionary waveform transformation approach

M. Caetano et al.

SBCM 2007

One can generate constantly changing sounds with optimization algorithms such as genetic algorithms: those algorithms tend to converge, but you can use them to model two populations, predators and preys, and have them *coevolve*.

***Applications of group theory
on granular synthesis***
R. Fabbri and A. Maia
SBMC 2007

Granular synthesis is an extreme case of additive synthesis: create a sound by combining (adding) hundreds or thousands of sound particles, defined as small areas or clouds in the time×frequency space (with a gaussian envelope and a width of 10 ms to 20 ms). The evolution of those grains with time is often defined by stochastic methods or finite automata. The article suggests to use group theory instead, with the symmetric group or some of its subgroups (alternating, dihedral and cyclic groups). The group action is not clearly defined; it could be:

- The action of the symmetric group on the set of all sound particles, not just those currently used;
- Some action on the parameters of the particles (frequency, duration, amplitude).

At each step, a random element from the group is selected to transform the set of particles.

The implementation relies on Sage, NumPy, PyAudiolib, Matplotlib, wxPython.

***Methods for evaluating clustering algorithms
for gene expression data
using a reference set of functional classes***
S. Datta and S. Datta
BMC Bioinformatics (2006)

How good are the results of your hierarchical clustering? In a biological context, you can use gene ontology (GO) databases to compute a biological homogeneity index and a biological stability index.

This could be generalized to a non-biological setup:

- For the homogeneity index, find another source of data and check if the clusters are consistent with it;
- For the stability, just resample and compare.

Also check the `pvclust` package.

Phantom probability
Y. Izhakian and Z. Izhakian
arXiv:0901.0902

To model the fact that the probability of an event, $P(A) \in [0, 1]$, is not known precisely, you can try to replace $[0, 1]$ by something else. I would have tried the ring of probability distributions on $[0, 1]$, but you can follow the lead of synthetic geometry, or algebraic geometry and its dual numbers, which adds a nilpotent element ε to the real line $\mathbf{R}[\varepsilon]/\langle \varepsilon^2 \rangle$, and add an idempotent (phantom) element \wp : $\mathbf{R}[\wp]/\langle \wp^2 = \wp \rangle$. Intuitively, use $a + b\wp$ to mean “I estimated the probability

as $a + \wp b$, where \wp is the probability of some other event on which I know nothing”. This can be generalized to

$$\frac{\mathbf{R}[\wp_1, \dots, \wp_n, \dots]}{\langle \wp_i \wp_j = \wp_{\text{Max}(i,j)} \rangle}$$

where the \wp_i are more and more hidden – you could have made them independent, $\mathbf{R}[\wp_1, \dots]/\langle \wp_i^2 = \wp_i \rangle$. Almost everything in probability theory (limit theorems, etc.) can be generalized to probabilities with values in a phantom ring.

Synthetic differential geometry
M. Schulman (2006)

Decision making in phantom spaces
Y. Izhakian and Z. Izhakian

Classical decision theory assumes that the probabilities of all events are known: it addresses risk (we do not know which event will be realized) but not uncertainty. With phantom probability theory, this can be generalized to situations where some but not all probabilities are known (each unknown probability is a nilpotent phantom probability). Plots of the efficient frontier have one more dimension for each phantom dimension.

***A survey of product-integration with a view
toward application in survival analysis***
R.D. Gill and S. Johansen
The annals of statistics (1990)

If X is a real (*i.e.*, the matrix coefficients are signed measures) matrix-valued measure on $(0, t]$, then its **product integral** is

$$Y(t) = \mathcal{P}_{s \in (0, t]} (\mathbf{1} + X(ds))$$

$$= \lim_{\substack{0=t_0 < t_1 < \dots < t_n=t \\ \text{Max}|t_i - t_{i-1}| \rightarrow 0}} \prod \left(\mathbf{1} + X((t_i, t_{i-1}]) \right)$$

It could also be defined by the Volterra equation (aka Kolmogorov forward equation)

$$Y(t) = \mathbf{1} + \int_{s \in (0, t]} Y(s^-) X(ds).$$

Here are some applications:

- Link between survival and hazard functions:

T : positive random variable (survival time)
 $S(t) = P[T > t]$ (survival function)

$$\lambda(t) = -\frac{d \log S(t)}{dt} \text{ (hazard rate)}$$

$$\Lambda(A) = \int_A \lambda(u) du \text{ (survival measure)}$$

$$S(t) = \mathcal{P}_{(0, t]} (\mathbf{1} - d\Lambda)$$

- In a censored context, the product-limit (Kaplan-Meier) estimator is the product integral of the empirical commulative hazard function (Nelson-Aalen estimator): this formulation can help derive properties of those estimators;

- For multivariate censored data, there is no classical estimator (the naive one would depend on a choice of coordinates) but the product-integral can provide a (complicated but) well-defined one;
- All this can be generalized to Markov processes (the survival analysis model uses 3 states: alive, censored, dead) by using matrices of size the number of states – it even works for countable state space Markov chains (e.g., Markov branching processes).

Current State of Java for HPC
B. Amedro et al.
Inria (2008)

Java is useable for high performance computing (HPC), e.g., with the ProActive library (LGPL, developed at the Inria) as an MPI replacement, but the performance greatly depends on the virtual machine (JVM) used and communications can be a bottleneck.

Versatile and declarative dynamic programming using pair algebras
P. Steffen and R. Giegerich
BMC Bioinformatics (2005)

When you need to implement several dynamic programming algorithms (they are everywhere in bioinformatics), you may want to factor out the algorithm into an algebraic dynamic programming (ADP) algorithm; what remains is then almost purely declarative:

- The alphabet used by the input sequence;
- The set of candidate solutions: they can be expressed as a set of trees, described by a *grammar*;
- A way to compute the score of a candidate tree (recursively).

Least angle regression
B. Efron et al. (2003)

An introduction to the Kalman filter
G. Welch and G. Bishop (2001)

Portfolio optimization in R
R in Finance 2009

Most portfolio optimization problems can be solved within R:

- For mean-variance optimization with no constraints (except perhaps a long-only one), use the `portfolio.optim` function in the `tseries` package;
- For the whole efficient frontier, call the function inside a loop;
- For the maximum Sharpe ratio portfolio, search on that efficient frontier;
- For mean-variance optimization with linear constraints (linear transaction costs, maximum weights, sectors or countries, etc.), use `solve.QP` in the `quadprog` package;
- Expected shortfall (ES, aka conditional value at risk, CVaR) optimization, is only a linear problem (but

beware: it is a *scenario-based* optimization) is just a linear problem: you can use the `Rglpk_solve_LP` function in the `Rglpk` package (it can also do mixed integer linear programming (MILP), but not quadratic programming);

- For more general constraints and/or performance/risk measures, such as

$$\Omega(r) = \frac{\int_r^\infty (1 - F(u)) du}{\int_{-\infty}^r F(u) du}$$

the drawdown (which can be plotted as an **underwater graph**, with the `chart.Drawdown` function), the **Rachev ratio** $R = \text{CVar}_\alpha^+ / \text{CVar}_\alpha^-$, you can check the `DEoptim` package.

A few remarks:

- Drawdown optimization looks very good in-sample, but it is a scenario-based optimization: how does it perform out-of-sample?
- There is no mention of constraints on the number of assets or of mixed integer quadratic programming (MIQP): what about `Rsymphony` (it is open-source)?
- There is no mention of stochastic programming – but it is not reasonable (yet?) for more than 2 or 3 assets.

Working with xts and quantmod
R in Finance 2009

The `xts` package relies on `zoo` to provide *time-based indexing*:

```
x <- xts(rnorm(N, Sys.Date()-1:N))
x["2009-01-01"]
x[index(x) >= Sys.Date()-5]
x["2009-01-01/2009-01-05"] # Interval
x["2009-01"] # One month of data
```

You may also want to check the `to.period` (convert to OHLC (open-high-low-close)), `endpoints` (to extract weekly, monthly, etc. data), `period.apply` functions.

The `quantmod` package abstracts data access, from local (RData, CSV, SQLite, MySQL) or public sources (Google, Yahoo, Oanda, FRED). The `getSymbols` functions assigns to local variables of the same name.

```
getSymbols("AAPL;IBM")
getSymbols("USD/EUR", src="oanda")
```

You may also want to check: `getDividends`, `getQuote`, `getSplits`, `getFX`, `getMetals`, `getOptionChain`.

There are also a few plotting functions.

```
candleChart(
  GSPC, subset="2009-01-01",
  theme="white", TA=NULL
)
chartSeries(IBM)
addBands()
?newTA # to add your own TA indicators
```

```
AddMACD(32,50,12)
reChart(
  subset="2009",
  theme="white", type="candles"
)
?chartSeries3d # For yield curves, etc.
```

Data can be loaded on-demand (lazily) and cached in memory or in a file.

```
attachSymbols(
  DB=DDB_Yahoo(),
  pos=2, prefix="E."
)
search()
str(ls("DDB:Yahoo"))
```

For more, check TTR, blotter, PerformanceAnalytics.

Statistics for Financial Engineering: Some R Examples R in Finance 2009

After clearly explaining the difference between linear, non-linear and non-parametric regression, the author presents three applications of statistics in finance.

The **probability of default** (PD) is often modeled as

$$P[\text{default} | \text{rating}] = \exp(\beta_0 + \beta_1 \text{rating});$$

which can be transformed via a **Box-Cox transformation** $p \mapsto (\kappa + p)^\lambda$ (unless the problem dictates another family of transformations); you cannot naively take the logarithm, because the probability can be zero (many textbooks do this, and have to throw observations away, resulting in a biased estimator). Whatever transformation you choose, do not forget the **residual analysis**. (The presenter wrote a book about transformations.)

Most stylized facts about an interest rate time series $(R_t)_t$ are visible in the following plots:

$$R_t \sim t, \quad \Delta R_t \sim t, \quad \Delta R_t \sim R_{t-1} \quad (\Delta t)^2 \sim R_{t-1}.$$

They can be modeled as a **diffusion process**

$$\Delta t = \mu(R_{t-1}) + \sigma(R_{t-1})\varepsilon_t$$

where the drift μ and the volatility σ can be non-linear (e.g., $\sigma(r) = \beta_0 r_1^\beta$) or **non-parametric**. With 50 years of data, non-parametric regression (e.g., with the **SemiPar** package – the presenter wrote a book on non-parametric estimation with the package author), non-parametric regression performs better: the parametric model will be outside the confidence interval of the non-parametric one. In this case, the residuals look GARCH

```
garch(x=res^2, order=c(1,1))
```

but the GARCH residuals still have some AR(1) noise

```
garchFit(
  formula = ~arma(1,0)+garch(1,1),
  data = res
)
```

and are far from Gaussian

```
garchFit(
  formula = ~arma(1,0)+garch(1,1),
  data = res,
  cond.dist = "std"
)
```

This 2-step process may be suboptimal.

The last example tries to predict future returns (starting with the worst):

- The average of the previous returns;
- The CAPM, forecast = $\beta \cdot$ previous returns;
- Bayesian shrinkage:

$$\text{forecast} = \lambda \cdot \text{previous returns} + (1-\lambda) \cdot \text{market returns}$$

- The previous market returns (this is the best forecast, but all stocks end up with the same forecast).

The **bayesian shrinkage** can be implemented with WinBUGS/R2WinBUGS (OpenBugs/BRugs is no longer on CRAN, and OpenBugs was written in a obscure, commercial, non-textual, Pascal-like language (the source file is a binary file); JAGS is not mentioned):

$$\mu_i \sim N(\alpha, \sigma_1^2) \\ r_{i,t} \sim N(\mu_i, \sigma_2^2);$$

this is a data-driven shrinkage: the amount of shrinkage is σ_1^2/σ_2^2 .

Risk Capital for Interacting Market and Credit Risk: VEC and GVAR models R in Finance 2009

International financial regulations (Basel II) assume that market and credit risk are independent and consider that assumption conservative: actually, the interaction between market and credit risk can be benign or malign. For small fluctuations, this can be seen from a Taylor expansion: let x be the risk factors (market, credit); let $f(x)$ be the value of your portfolio, then

$$f(x + \varepsilon) = f(x) + f'(x) \cdot \varepsilon + f''(x) \cdot \varepsilon \cdot \varepsilon + o(|\varepsilon|^2),$$

where the bilinear form $f''(x)$ can be positive, negative, or neither. This shows the problem for small fluctuations: we would like a decomposition

$$f(x + \varepsilon) = f(x) + g_{\text{market}}(x, \varepsilon_{\text{market}}) + g_{\text{credit}}(x, \varepsilon_{\text{credit}})$$

for large fluctuations as well – this works iff the portfolio is *separable*, i.e., $f(x) = f_{\text{market}}(x) + f_{\text{credit}}(x)$.

The rest of the presentation is a confusing example (the GVAR in the title is apparently a vector autoregression (VAR) with fewer parameters).

ICA for multivariate financial time series R in Finance 2009

Since independent component analysis (ICA) is an optimization problem, you can easily add constraints; you could define a **constrained PCA** in the same way.

This presentation also tries to give technical details on how ICA is estimated (but it quickly becomes a list of formulas with no explanations) and how a time dimension can be added (but their “time-varying mixing matrix” looks like a moving window estimator).

Random portfolios: Practice and Theory R in Finance 2009

Random portfolios can be seen as a kind of bootstrap or permutation test.

The basic idea is the following: since we add so many constraints, many things only depend on the constraints; random portfolios can measure the performance or risk coming from the constraints and isolate the value added by the portfolio manager.

Random portfolios (and matched portfolios) can replace a benchmark or a peer group: they will be less biased and lead to tests with a higher **power**; they can also incorporate more information, such as the position at the beginning of the period and the turnover constraint.

Random portfolios can also help measure the (sometimes unwanted) impact of constraints: you can compare the performance and risk of random portfolios for several sets of constraints (e.g., loose or tight constraints on sector weights). [This was new.]

Generating random portfolios is problematic. [This is also new.]

- A rejection strategy is not reasonable (there are too few portfolios satisfying the constraints);
- Algorithms to sample inside a polytope do not apply if there are non-linear constraints;
- You can compute an optimal portfolio for random returns, but this may be biased;
- You can use a random search: start with a random, unconstrained portfolio and minimize a penalty for broken constraints; however, you will be close to a binding constraint (this does not worry me: we are in high dimension, everything is close to the boundary) and the sampling will not be uniform (this is not very worrying: this is almost exactly what a portfolio manager does, except that he starts with non-random returns).

Matching portfolios D. Kane R in Finance 2009

To measure the performance of a portfolio, one may look at the average capitalization, the momentum, and the book market value of its constituents and compare with the corresponding quintile portfolio: but you have a single portfolio and the more factors you add,

the emptier those quintile portfolios become. *Matching* views those portfolio characteristics (except those in which the portfolio manager claims expertise) as constraints and build a portfolio matching those constraints.

I do not see the difference with *random portfolios*.

Quantile regression in R R in Finance 2009

The definition of a quantile can be formulated as a linear optimization problem and this generalizes readily to a regression setup

$$\alpha_t = \underset{\alpha \in \mathbf{R}}{\operatorname{Argmin}} \sum_{i=0}^n \rho_{\tau}(y_i - \alpha)$$
$$\rho_{\tau}(u) = (\tau - \mathbf{1}_{u < 0})u.$$

Even with a linear model, quantile regression can highlight non-linear features, such as the conditional distribution of the data (this is not as general as the distribution of $y|x$, but almost: there is a model, which assumes a linear relation between the quantiles of x and (a linear combination of) y , but that relation can depend on the quantile; in particular, we can end up with non-gaussian, and even multimodal, distributions) or the dependence of the regression coefficients on the quantile (for an additive gaussian model, we would expect the intercept to be the inverse cumulative distribution function of the quantile and the other coefficients to be constant).

Two plots: conditional densities (this is similar to a violin plot, but with an underlying linear model); coef \sim quantile.

In a time series setup, you can use quantile regression to estimate a **quantile autoregression** (QAR): it will be non-gaussian, but still partly parametric.

Quantile regression is actually equivalent to CVaR (conditional value at risk) portfolio optimization; by approximating the weight function of a spectral risk measure by a locally constant function, quantile regression can be used to optimize portfolios for any “pessimistic” risk measure (this has nothing to do with quantile regression: it applies to linear programming in general).

The presentation ends up with a non-gaussian regression

$$\text{score} \sim \text{team 1} + \text{team 2} + \text{home advantage}$$

(all the predictors are qualitative variables, the variable to predict is a count variable: this would classically be estimated by a Poisson model – but we would have to check this distributional assumption). With quantile regression, we keep the linearity of the model, but remove any distributional assumption; the model can be estimated as a (not unreasonably) large but sparse linear program and one can *sample* from the distribution of scores, in a half-bayesian fashion.

Quantile regression can be seen as a prior-less bayesian regression with no distributional assumptions.

Using R for hedge fund of funds management
E. Zivot
R in Finance 2009

When faced with non-gaussian data, use the Cornish-Fisher expansion to compute the value at risk (VaR) or the expected shortfall (ES, CVaR).

If the time series of returns do not have the same length: select the best risk factors; compute the exposure of the funds to those factors; simulate the missing returns; use those simulated returns to estimate the marginal contribution to risk (MCTR) or risk attribution for VaR or ES. This is available in the **PerformanceMetrics** package.

This reminds me of the problem of estimating a variance matrix $\text{Var}(x_1, \dots, x_n)$ when there are many missing values: the maximum-likelihood estimator can be computed explicitly (this is “just” linear algebra, albeit horrible), but an MCMC simulation is easier to implement (and less likely to be buggy). However, even for toy examples, the convergence is horribly slow.

The presentation mentions but does not tackle the following problems:

- Biases (survivorship, reporting, backfill);
- Serial correlation (performance smoothing; illiquid positions).

Particle learning
R in Finance 2009

Not understandable. None of the articles they refer to are available.

Performance Analysis in R
R in Finance 2009

Clear presentation of what the **PerformanceAnalytics** package can do: measure, compare, decompose the performance or risk of portfolios.

Market Microstructure Tutorial
R in Finance 2009

Market microstructure is the study of why markets are not efficient and how to profit from it (or, equivalently, help make them more efficient); this includes behavioural effects.

The *Glosten-Milgrom model* considers two traders: one (informed) always buys (or sells, this is fixed at the beginning); the other (non-informed) buys or sells with probability 1/2 (they just buy or sell, the model does not consider the need to find a counterparty); only one trader (selected at random) trades at each point in time. We define the *bid* (respectively, *ask*) as the probability that the informed trader is buying (resp. selling) given that the last trade was a buy (resp. sell). The bid and ask converge towards the true price (0

or 1) (*i.e.*, the market goes where the informed trader wants: market manipulation is indistinguishable from price discovery).

The *Kyle model* considers a market maker facing informed and non-informed traders and sets the price to the expected true price given the total order size he sees; the informed trader chooses the order size to maximize profit. There again, information leaks very quickly.

Event study: a change-point model approach
R in Finance 2009

Hidden variable (aka latent variable, or regime switching if that variable is discrete) models can be estimated via Monte Carlo simulations. In this example, we consider spells of iid gaussian random variables

[[PLOT]]

and estimate the model by Gibbs sampling, with a bayesian prior:

- Estimate the state at each point in time;
- Estimate the parameters in each spell;
- Estimate the transition probabilities;
- Iterate.

(At each step, we consider the whole time series.) This can be repeated with a different number of regimes, and the best model can be selected by looking at the *Bayes factor*.

In a financial setting, breakpoints need not coincide with announcements: they can occur before.

Detecting structural breaks in tail behavior
W.H. Liu
R in Finance 2009

The *Hills estimator* of the tail index (and the tail index itself) only makes sense for power law tails and, even in that case, is known to be very noisy. Replacing the power law by a general extreme value (GEV) distribution does not solve those problems: tests for structural breaks (I do not know them: SupLN, OLS-based CUSUM, Nyblom-Hansen, generalized M-fluctuation) return a surprisingly high number of potential regime changes.

For risk management purposes, estimators of GEV models are as problematic as the Hills estimator.

Portfolio Optimization with R/RMetrics
R in Finance 2009

Quick overview of what is available in **RMetrics** for portfolio optimization – there is an ebook with the same title (it excludes: quadratic (non-linear) constraints, integer constraints, non-linear objectives, Black-Litterman copula pooling (BLCP), quadratic lower partial moments (QLPM), copula tail risk: these will be covered in a second volume). The list of packages looks exhaustive: **Rglpk**, **Rsymphony**, **Rlpsolve**, **quadprog**, **Ripop**, **Rsocp**, **Rdonlp2**, **Rnlminb**.

Econometrics and practice: mind the gap!
Y. Chalabi and D. Würtz
R in Finance 2009

There is an unreasonable number of variants of the GARCH model, but fitting them poses many practical issues (bugs, parameter initialization, optimization schemes, etc.).

The presentation slides contain some sample code: it is surprisingly straightforward (it would be trickier to estimate a stochastic volatility model, *i.e.*, a model with two sources of randomness: you would need to consider a time series of (hidden) innovations).

The stationary assumption is often violated.

We might be tempted to use *robust* methods to throw outliers away, but we cannot afford it: we would underestimate risk. Instead, you can include them in the model, either as isolated points or as a different regime: **MS-GARCH** (Markow-Switching GARCH) is one such example.

Indirect inference can help fit a difficult-to-fit model g by using an (easier-to-fit) model f :

- Estimate the auxiliary model f in the data y : $\hat{\theta}$;
- For all possible values of the parameters ρ of the model g , generate a sample \tilde{y} , and estimate the auxiliary model f on it: $\tilde{\theta}(\rho)$.
- Find the parameters ρ that minimize the distance $\|\hat{\theta} - \tilde{\theta}(\rho)\|$.

Introduction to high-performance computing with R
D. Eddelbuettel
R in Finance 2009

You can profile your code with the `system.time`, `Rprof`, `Rprofmen`, `tracemem` functions; the `profr` and `proftools` package; the `prof2dot.pl`, `valgrind`, `kcachegrind`, `sprof`, `oprofile`, Google PerfTools tools.

Vectorization, with the `*apply` or `outer` functions, or just-in-time compilation with Ra (a set of patches for R, available for Debian/Ubuntu – a very search-engine-unfriendly name) and the `jit` package, can speed things up.

For the best performance, you can rewrite the most time-consuming parts of the code in C/C++: with `.C`, the C function uses C types; with `.Call`, it uses R types (`SEXP`); with `Rcpp`, the conversion between R and C++ types (templates) is transparent. The `inline` package and the `cfunction` function allow you to put the C code (with `.Call` or `.C` conventions) directly inside the R code – as with Perl's `Inline::C` module.

In the other direction, `RInside` allows you to evaluation R code from a C/C++ program.

Other high-performace topics were not mentionned:

- Parallel computing: `MPI`, `nws`, `snow`;
- Out-of-memory computations: `biglm`, `ff`, `big-`

- `matrix9`;
- Scripting and automation.

A latent-variable approach to validate credit rating systems with R
B. Grün et al.
R in Finance 2009

Credit rating of several firms (obligors) by several raters (banks, rating agencies) can be modeled as a mixed model: let the observed score $S_{i,j}$ be the probit of the announced probability of default for obligor i according to rater j :

$$S_{i,j} = S_i + \mu_j + \sigma_j Z_{i,j},$$

where S_i is the latent (consensus) rating and μ_j the bias of rater j .

This can be generalized into a dynamic model by assuming the latent probability is AR(1) and examined, by Gibbs sampling, with the `rjags` and `coda` packages. In particular, you can look at: bias and variance of raters, consensus rating; and the residuals can be further analyzed.

In practice, you have to map the ordinal rating of the rating agencies (AAA, etc.) into probabilities: this hides the bias.

RMeCab

MeCab is a Japanese morphological analyzer, *i.e.*, a tool to cut sentences into words (there are no spaces between words in Japanese). It looks more robust than its competitors (ChaSen, CaboCha) and can now be used from R.

After installing R, MeCab, MeCab-ipadic (beware, they apparently only provide a *binary* package, even for Linux):

```
install.packages(
  "RMecab.tar.gz",
  destdir=".", repos=NULL
)
library(RMecab)
```

To split a sentence into words, and get the POS (part-of-speech) of those words:

```
RMecabC("猫が鼠を食べた。")
```

The default output of the command-line tool (word, POS0, POS1, POS2, POS3 (more and more detailed), conjugated form (?), dictionary form, pronunciation, pronunciation (yes, twice)):

```
RMecabText("text.txt")
```

The frequency dataframe, with primary key word-POS0-POS1 (sort it before use):

```
RMecabFreq("text.txt")
```

The term-document matrix:

```
docMatrix("corpus/")
d <- docMatrix(
  "corpus/",
  pos=c("名詞", "形容詞"))
cor(d)
```

N-grams:

```
Ngram("text.txt", N=4)
```

However, my current (NLP) interest is lexicography and I am skeptical about the ability of R/MeCab to handle very large datasets (say, all of Wikipedia).

I still have to read the book, *Rによるテキストマイニング入門* (石田基広, 2008).

Pessimistic Portfolio Allocation and Choquet Expected Utility

Details on the relation between

- Choquet expected utility (very similar to *cumulated prospect theory*: when computing the expected utility, we give more or less importance (we distort the probability of) favourable or unfavourable events);
- coherent (or spectral, or pessimistic) risk measures.

Estimation and decomposition of downside risk for portfolios with non-normal returns

K. Boudt et al (2007)

Some people are apparently still using gaussian parametric value at risk (VaR) estimators...

After recalling the Edgeworth expansion of a cumulative distribution function (around the Gaussian)

$$G(z) = \Phi(z) + \sum_{i=1}^r P_i(z)$$

and the Cornish-Fisher (CF) expansion of the corresponding quantile function

$$G^{-1}(\alpha) = z_\alpha + \phi(z_\alpha) \sum_{i=1}^r P_i^*(z_\alpha)$$

$$z_\alpha = \Phi^{-1}(\alpha)$$

the authors compute the **modified VaR** (mVaR) and the **modified expected shortfall** (mES), corresponding to the second order Cornish-Fisher expansion, *i.e.*, a Gaussian distribution modified to have non-trivial skew and kurtosis.

By differentiating the mVaR or mES with respect to portfolio weights, one can compute risk attribution.

Since the estimation of higher moments is noisy, the authors suggest to clean the data before computing skewness and kurtosis: they do not remove or truncate outliers, but just *shrink* them (the cleaning threshold should be beyond the desired VaR threshold).

SemiPar, an R package for semiparametric regression

M.P Wand et al. (2005)

The main R functions for semiparametric regression are `gam::gam`, `SemiPar::spm`, `mgcv::gam`; you may also want to check the `lmeSplines` and `fields` packages. `SemiPar` is younger than `gam`, but allows for bivariate terms and better plots.

```
r <- spm( y ~ f(x) )
```

```
plot(r)
points(x, y)
plot(r, drv=1) # First derivative
```

You can impose the smoothing parameter (e.g., `spar=3`), the number of degrees of freedom (`df` for a single term, `adf` (approximate degrees of freedom) if there are several terms) or the basis (e.g., `basis="trunc.poly"`, `degree=5` – by default, cubic radial basis functions).

```
u ~ x + f(y)      # semi-parametric
u ~ f(x) + f(y)    # additive model
u ~ f(x,y)         # bivariate smoothing
```

For bivariate smoothing (used in spatial statistics, you may want to check and/or alter the nodes: there may be too few or too many). For higher dimensions, you can use a *geoadditive* model

```
u ~ f(x,y) + f(z)
```

The function also accepts binary data (`family="binomial"`), count data (`family="poisson"`), or mixed models (with the usual horrible syntax).

Unit root quantile autoregression inference

R. Koenker and Z. Xiao

Financial time series are often autoregressive with a root near unity: for quantile auto-regression, this changes the asymptotic distributions, but the correct tests are more powerful (for non-gaussian innovations) than tests based on a gaussian model.

Conditional quantiles of volatility in equity index and foreign exchange data

J.W. Galbraith (2001)

Quantile regression $r_t \sim r_{t-1} + \sigma_t$ can highlight asymmetric behaviour in time series of foreign exchange returns.

VAR, SVAR and SVEC Model: Implementation within R package vars

B. Pfaff

Journal of Statistical Software (2008)

The `vars` package estimates vector autoregressive (VAR) models

$$y_t = A_1 y_{t-1} + \dots + A_p y_{t-p} + u_t$$

or structural VAR (SVAR) models

$$A y_t = A_1^* y_{t-1} + \dots + A_p^* y_{t-p} + B \varepsilon_t$$

(for the model to be identifiable, you have to add restrictions (prior knowledge) on A and/or B) or structural vector error correction (SVER) models (not explained clearly). The package should be used in conjunction with the `urca` package, whose author systematically refuses to output p -values and only provides test statistics and a couple of critical values.

***Generalized linear mixed model analysis
via sequential Monte Carlo sampling***
Y. Fan et al. (2007)

Generalized linear mixed models (GLMM) are notoriously more complicated to estimate than linear mixed models or generalized linear models. **Sequential Monte Carlo** (SMC) sampling is an alternative to Monte Carlo Markov Chain (MCMC) sampling: it is a *static analogue of particle filters*. You have to add a time dimension to the problem, for instance, by adding the observations one by one (the time-indexed family of posterior distributions is more and more precise) or by sampling from the family of posterior distributions $(\pi_0^{1-t}\pi^t)_{t \in [0,1]}$, where π is the desired posterior and π_0 the prior or a gaussian distribution centered on an approximate fit.

***Parsimonious classification
via generalized mixed models***
G. Kauermann et al. (2007)

Supervised classification can be performed via a generalized additive logistic model (logistic GAM)

$$P(Y = 1) = f_1(x_1) + \dots + f_p(x_p)$$

where the unknown functions f_i are estimated as generalized splines or, equivalently, as a mixed model

$$f_i(x) = \sum_j u_{i,j} b_j(x)$$

$$u_{ij} \sim \text{some prior}$$

where the b_j are the basis functions and the u_j their coefficients.

See also the `gam.step` function for a similar approach.

***Anomalous returns
in a neural network equity-ranking predictor***
J.B. Satinover and D. Sornette (2008)

A nonlinear AR(10) model fitted on quarterly retruns with a neural returns can beat the S&P 500 (or would have, in the 1990s), even if you include trading costs (but with a much higher volatility).

***Detecting speculative bubbles
created in experiments
via decoupling in agent-based models***
M. Roszcynska et al. (2008)

Bubbles are periods when everyone makes the same investment decisions (one could argue that short-selling helps prevent bubbles, by allowing people to make opposite investment decisions).

Decomposition of proper scores
J. Bröcker (2008)
arXiv:0806.0813

Scores are a way of measuring the quality of estimators of probabilities (or estimators of probability distributions).

An estimator \hat{p} of the probability of a binary random variable Y is *reliable* if

$$\forall p \quad P[Y = 1 \mid \hat{p} = p] = p.$$

Given an observation $y \in [0, 1]$ of Y , the **Brier score** of \hat{p} is $(y - \hat{p})^2$; with several observations and forecasts, you would take the average of those quantities. This sum of squares is used in decision theory.

This can be generalized to non-binary (discrete or continuous) random variables (the article is not always clear); some properties of the Brier score lead to its decomposition into *reliability* and *sharpness* (information content).

Model of information diffusion
D.V. Lande

Cellular automata (the game of life) on a 2-dimensional grid can be used to model information diffusion. (I am skeptical about the lack of influence of the graph structure: a 2-dimensional graph and a small-world graph are very different – physicists are already used to seeing differences between 2- and 3-dimensional phenomena...)

***“Bubbles in society”
The example of the United States
Apollo program***
M. Gisler and D. Sornette

Bubbles, *i.e.*, collective over-enthusiasm, are (unavoidable and) beneficial: they foster innovation and allow the development of new technologies; examples include wars, cold wars, space programs, etc.

***The w-index:
a significant improvement on the h-index***
Q. Wu (2008)

One can reduce the number of citations papers of a researcher get (or any finite sequence of positive numbers) to a single index,

$$\sup\{w : \text{he has } w \text{ papers cited at most 10 times}\}.$$

If you plot the number of citations against the paper number w (after ranking the papers in decreasing citation order), this is the intersection of the curve with the $n = 10w$ line. Of course, this “10” is arbitrary...

***Probing the improbable:
methodological challenges for risks
with low probabilities and high stakes***
T. Ord et al.

When estimating very low probabilities (e.g., asteroid impact), you should consider that the argument could be incorrect, for many reasons:

- Theory (e.g., choice of Newtonian mechanics or general relativity);

- Model: is it broad/detailed enough (which celestial bodies to include, influence of their shape or structure, etc.); retraction rates of academic papers suggests that the proportion of inadequate models is at least 10^{-3} ;
- Calculations: intentional simplification (if a physicist is doing the computations) or errors: the probability of computational errors varies between 1% (drug doses in hospitals) and 90% (if you use spreadsheets).

(The problem is sometimes obvious in the conclusion: claims that something “is impossible” are almost always wrong.)

The estimate can be improved by providing several models.

Contrary to what the title suggests, the article only focuses on the probabilities, not on the stakes: there can be unpleasant situations where $\int \text{Stake } dP = \infty$.

Nonlinear dimension reduction with kernel sliced inverse regression

Y.-R. Yeh et al. (2007)

The sliced inverse regression (SIR), a linear dimension reduction technique (not explained clearly: in a regression setup $y \sim x$, look for $y = f(\beta x)$, where βx has much fewer columns than x and f is a linear or non-linear function, by looking at $\Sigma_{x|y}$ and Σ_x) can be extended with the “kernel trick”.

The Fast- τ estimator for regression

M. Salibian et al.

A τ -estimator is defined as

$$\hat{\beta} = \underset{\beta}{\text{Argmin}} s^2(\beta) \frac{1}{n} \sum_i \rho_2 \left(\frac{y_i - \beta x_i}{s(\beta)} \right)$$

$$\frac{1}{n} \sum_i \rho_1 \left(\frac{y_i - \beta x_i}{s(\beta)} \right) = 1.$$

(The second equation defines s .) If $\rho_1 = \rho_2$, it is an S -estimator (and the first equation simplifies to $\hat{\beta} = \underset{\beta}{\text{Argmin}} s(\beta)$). The *breakdown* is determined by ρ_1 ; the *efficiency* by ρ_2 .

Contrary to common belief, τ -estimators can be efficiently implemented.

Robust performance testing with the Sharpe ratio

O. Ledoit and M. Wolf

Journal of empirical finance (2008)

The tests traditionally used to compare Sharpe ratios are not robust to fat tails, autocorrelation or volatility clustering. One can use the same test statistics, but “assume” that their limiting distribution is $N(0, \Psi)$, with Ψ unknown (instead of known) and estimated with an HAC (heteroskedasticity and autocorrelation robust) estimator. The resulting p -values are higher, *i.e.*, apparent differences are less significant.

Correlation, hierarchies and networks in financial markets

M. Tumminello et al.

arXiv:0809.4615

Hierarchical clustering from a correlation matrix (minimum spanning tree (MST), average linkage MST, etc. – try several algorithms) can also be used to *filter* the correlation matrix: if C_1 and C_2 are two separate subtrees joined by an edge (or an internal node),

$$\exists c \quad \forall i \in C_1 \quad \forall j \in C_2 \quad \text{Cor}(i, j) = c$$

(as a result, there are as many (potentially) different values in the correlation matrix as internal nodes: $n - 1$). The structure of the filtered correlation matrix can replace the traditional factor model $V = \text{eve}'$. This **hierarchical nested factor model** has one factor for each internal node, but a *bootstrap* will often suggest that only a few are significant: fuse the non-needed nodes with their parent.

To check the goodness-of-fit of this hierarchical nested factor model, compare the Kullback–Leibler distance between the filtered matrix and the (raw) observed one with what you would expect of the observed matrix if the filtered matrix was the right one (with a multivariate Gaussian or Student distribution).

To check the stability of the filtering algorithm, check the average Kullback–Leibler distance between filtered bootstrapped matrices: random matrix theory (RMT) is bad, shrinkage and hierarchical nested factor models are good.

Stock market volatility:

an approach based on the Tsallis entropy

S.R. Bentes et al.

arXiv:0809.4570

Entropy is often used in a cross-sectional way, on the distribution of prices (or market capitalizations) of stocks at a given date, but you can also use it in a time series fashion, on the distribution of the returns of a single stock, as yet another measure of volatility.

They authors fail to properly define and motivate the Tsallis entropy

$$S_q(p) = \frac{1 - \int p^q}{q - 1}$$

they prefer over the Shannon entropy

$$H(p) = \int -p \ln p.$$

Smile dynamics:

a theory of implied leverage effect

S. Ciliberti et al. (2008)

arXiv:0809.3375

The volatility surface, if you only want a Taylor expansion around the money, can be studied without a complete model of the returns of the underlying (*e.g.*,

local volatility (the volatility is a deterministic function of the price), stochastic volatility, GARCH, jumps and Lévy processes, multiscale fractals, SABR, etc.): the *term structure* of the skewness (and higher moments) suffices

$$\Sigma(K, T) = \sigma \left(1 + \frac{\zeta(T)}{6} M + \frac{\kappa(T)}{24} (M^2 - 1) + O(M^3) \right)$$

$$M = \frac{1}{\sigma\sqrt{T}} \left(\frac{K e^{-rT}}{S} - 1 \right)$$

where K is the strike, T the maturity, M the money-ness, $\zeta(T)$ the skewness (of the returns of the underlying over intervals of size T), $\kappa(T)$ the kurtosis. The *term structure* of the skewness

$$\zeta(T) = \frac{\zeta(1)}{\sqrt{T}} + \frac{3}{\sqrt{T}} \sum_{t=1}^T \left(1 - \frac{t}{T} \right) \rho_{\text{leverage}}(T)$$

can be computed from the *leverage correlation function* (that quantifies the *leverage effect*, *i.e.*, the correlation between past returns and future squared returns)

$$\rho_{\text{leverage}}(t) = \frac{E[r_i r_{i+t}^2]}{\sigma^3}$$

which can be approximated as

$$\rho_{\text{leverage}}(t) = -A e^{-t/\tau}.$$

The *dynamics* of the implied volatility are between those suggested by the *sticky strike* ($\Sigma = \Sigma(K, T)$ (does not change as time passes), an upper bound, correct for short maturities) and the *sticky delta* ($\Sigma = \Sigma(M, T)$, lower bound, good approximation for longer maturities).

In the same way a 1-factor model of the returns of a stocks allows the volatility to be decomposed into a market volatility and an idiosyncratic volatility, it also allows for the decomposition of the skewness (into three terms).

Similarly, long-range volatility clustering would induce a non-trivial term structure on the kurtosis.

***An extensive set of scaling laws
and the FX coastline***
J.B. Glattfelder et al.
arXiv:0809.1040

The laws are all of the form something $\propto (\Delta x)^k$ or something $\propto (\Delta t)^k$ where Δx is the price change (that could be interpreted as an “intrinsic time”), Δt a time interval and “something” is the L^p average ($p \in \{1, 2\}$) of:

- Absolute returns in Δt ;
- Number of ticks in Δt , or Δx ;
- Number of direction changes in Δt , or Δx ;
- Maximum price move (drawdown) in Δt ;
- Waiting time for a price move of Δx (in absolute value or not);
- Total variation in Δt , or Δx ,

***Solvable stochastic dealer models
for financial markets***
K. Yamada et al.
arXiv:0807.0481

Yet another agent-based model, whose agents have different horizons and try to predict future prices with moving averages.

***Bayesian analysis of value-at-risk
with product partition models***
G. Bormetti et al.
arXiv:0809.0241

Empty article suggesting to use gaussian mixtures (instead of a gaussian model) to estimate value-at-risk (VaR). (“Product partition” is the physicaese word for “gaussian mixture”).

***Minimal agent based model
for financial markets I:
origin and self-organization of stylized facts***
V. Alfi et al.
arXiv:0808.3562

Stylized facts could be a *finite size effect*: they could disappear as the number of agents grow.

The authors build their (uninteresting) model on:

- Short-horizon investors;
- Herding
- The only available information is the price time series, ie, everyone is a chartist, with either a long or short horizon, and looks at what others are doing.

***Towards a common framework
for statistical analysis and development***
K. Imai et al.
**Journal of computational and graphical
statistics (2008)**

What about “refactoring statistical analysis” (or, equivalently, on “ontology of statistical analysis”) in the way you refactor (statistical) software? The article advocates the **zelig** package (a unified interface to models implemented in other packages), explains how R’s formulas can be generalized to accomodate constants and more complex models, presents Zelig’s dynamically-generated GUI and the *dataverse network* (a data-sharing/publishing platform).

***Efficient emulators for multivariate
deterministic functions***
J. Rougier (2007)

Climate models can be seen as (deterministic) functions $f : \mathbf{R}^N \rightarrow \mathbf{R}^M$, $x \mapsto y = f(x)$ with N very large. Since they are expensive to evaluate, they are often replaced by statistical approximations or *emulators*, *i.e.*, regressions $y \sim x$ from a small number of evaluations, where both x and y are vectors and have a spatial dependency structure modeled by a gaussian process.

In case the output is a vector and you are more familiar with scalar outputs, you can:

- Forecast the coordinates independently
- Perform a principal component analysis (PCA) or, better, a canonical correlation analysis (CCA) on the output space;
- uncurrify the function to be evaluated, *i.e.*, replace $x \mapsto (y_1, \dots, y_n)$ by $(x, i) \mapsto y_i$ (this assumes that the y_i are commensurate);
- Replace $x \mapsto f(x)$ by $x \mapsto N(\hat{f}(x), V(x))$ where $\hat{f}(x)$ is the predicted value and $V(x)$ its precision.

The article explains how to use linear algebra to approximate and speed up the estimation of this gaussian process.

Residual-based shadings for visualizing (conditional) independence
A. Zeileis et al.

Mosaic plots (that display contingency tables) and *association plots* (that display the Pearson residuals of a log-linear model $r_{ij} = (n_{ij} - \hat{n}_{ij})/\sqrt{n_{ij}}$) can be augmented by shadings or colours to display the significance of (*i.e.*, the p -value of a statistical test for) departure from independence:

- The usual χ^2 of independence, $\chi^2 = \sum_{ij} r_{ij}^2$ can be replaced by $M = \text{Max}_{ij} |r_{ij}|$, so that the whole test is significant iff a cell is significant;
- The critical values of this test (used to decide whether to colour or not) are data-dependent: they can be computed by *permutation tests*;
- The traditional colourings rely on the HSV (hue, saturation, value) *colour space* (a cone, whose vertex is black, whose base has a fully saturated rim and a white center) and can be misleading (saturation is not uniform accross hues); it can be replaced by the HCL (hue, chroma, luminance) space (a double cone, with a white vertex on one side and a black vertex on the other, and a fully saturated rim in the middle – that rim is slightly broken: admissible chroma and luminance values depen on the hue);
- Instead of changing just the saturation, you can also change both chroma and luminance;
- When there is nothing significant, use a medium grey (not white); as significance increases, increase the range of hues (a more colorful picture looks more significant).

Those ideas are implemented in R in the `vcd` package: check the `mosaic`, `assoc` and `cotabplot` functions.

Singular value decomposition and its visualization
L. Zhang et al. (2005)

To study a periodic signal (of known periodicity, *e.g.*, for hourly data you would expect a daily pattern), you can put the data in a matrix (dates in rows, time in columns – you can already plot this matrix as a surface to see intraday and interday patterns) and perform a singular value decomposition (SVD): it expresses the

matrix as a sum of rank-1 matrices. Plotting the data corresponding to matrices 1, 1+2, 1+2+3, etc. can highlight patterns in the data. Potential applications are similar to those of functional data analysis (FDA).

Delineation of irregularly shaped disease clusters through multiobjective optimization
L. Duczmal et al.
Journal of computational and graphical statistics (2008)

To detect a spatial cluster, in epidemiology, compute the *Kulldorff scan statistic*

$$\frac{\text{incidence inside}^{\text{cases inside}}}{\text{incidence outside}^{\text{cases outside}}}$$

for each (connected) potential cluster and select the one with the highest values (for circular clusters, this statistic is well approximated by a *Gumbel distribution*). This will select non-significant tree-like “clusters”: you can add a penalty, such as the *compactness*

$$\frac{\text{Area}}{\text{Parameter}^2}$$

and maximize

$$\left(\frac{\text{incidence inside}^{\text{cases inside}}}{\text{incidence outside}^{\text{cases outside}}} \right) \left(\frac{\text{Area}}{\text{Parameter}^2} \right)^\alpha$$

for some value of α (higher values favour circular shapes, while the actual cluster may be around a road, river, shore or plume or air pollution).

Instead, one can use a genetic algorithm to find clusters in the scan statistic vs compactness *Pareto set* (*i.e.*, efficient frontier). (Most machine learning algorithms can be generalized to multi-objective optimization.)

Modeling spatial-temporal binary data using Markov random fields
J. Zhu et al.

The spatial linear model

$$x_{i,j} \sim y + x_{i+1,j} + x_{i-1,j} + x_{i,j-1} + x_{i,j+1}$$

can be generalized for spatio-temporal data

$$x_{i,j,t} \sim y + x_{i+1,j,t} + x_{i-1,j,t} + x_{i,j-1,t} + x_{i,j+1,t} + x_{i,j,t-1}$$

The article does this in the context of a binary (autologistic) model (infected or not) and suggests to estimate it properly (previous attempts were estimating the spatial and temporal components separately), via Gibbs sampling.

Autologistic regression analysis of spatial-temporal binary data via Monte Carlo maximum likelihood
J. Zhu et al. (2008)

Terser but illustrated article on the spatio-temporal autologistic model

Hierarchical structure and the prediction of missing links in networks

A. Clauset et al.
arXiv:0811.0484

A **hierarchical random graph** is build from a dendrogram with a probability assigned to each internal node, by linking two vertices with the probability coming from their closest common ancestor.

Conversely, one can use a maximum likelihood (MCMC) approach to find the hierarchical random graph closest to a given network.

Once found, this hierarchical organization can also be used to predict missing connections.

Statistical properties of information flow in financial time series

C. Eom et al.
arXiv:0811.0448

The information flow between stocks can be studied by looking at

- the Granger causality within all pairs of stock retruns.
- the minimum spanning tree of the correlation matrix of their returns.

Serial corelation and heterogeneous volatility in financial markets: beyond the LeBaron effect

S. Bianco et al.
0810.4912

The *LeBaron effect*, *i.e.*, the negative correlation between volatility and autocorrelation,

$$\langle x_t^3 x_{t-1} \rangle < 0,$$

exists in high-frequency data.

Look-ahead benchmark bias in portfolio performance evaluation

G. Daniel et al.
arXiv:0810.1922

If you are assuming that the composition of your benchmark (often, an index) is constant and use the constituents from the end of the test period, you have a textbook example of a *look-ahead bias* and/or *survivor bias* (many indices are defined as the top N stocks in some market). The authors measure that bias: it can be as large as 8% per annum for the S&P 500.

Volatility effects on the escape time in financial market models

B. Spagnolo and S. Valenti
arXiv:0810.1625

One can use the *probability distribution of escape times* to analyze (or build a more formal statistical test of) the goodness of fit of a financial time series model.

Scale free effects in world currency network

A.Z. Górski et al.
arXiv:0810.1215

Yet another article on the minimum spanning tree of the correlation matrix of exchange rates; this one also looks at what happens if one changes the reference currency.

From time series to complex networks: the visibility graph

L. Lacasa et al.
arXiv:0810.0920

Here is an algorithm to turn a time series into a graph (feel free to imagine yours and check if/how time series properties translate into graph properties): one vertex for each observation, two vertices are linked if one can “see” the other in the flatland world `plot(.,type='h')`. A periodic time series becomes a random graph; a fractal time series becomes a scale-free network.

Portfolio optimization under habit formation

R. Naryshkin and M. Davidson
arXiv:0810.0678

The classical 1-period portfolio optimization problem

$$\text{Maximize } E[U(W_T)]$$

(maximize the expected utility of final wealth) can be replaced by the (also classical) multiperiod (or continuous time) optimal investment/consumption/bequest problem

$$E \int_0^T e^{-\rho t} U(C_t) dt + e^{-\rho T} B(W_T)$$

(maximize the discounted utility of consumption and utility of bequest, where T is the terminal time, C_t the consumption, W_t the wealth, U the utility of consumption, B the utility of bequest). This model has several drawbacks: the utility of consumption remains constant (or predefined). it should actually depend on the previous consumption

$$\bar{C}_t = \frac{1}{t} \int_0^t C_u du$$

to account for habit formation or intolerance to a decline in the standard of living.

The article suggests a utility of the form

$$U = T \left(\frac{C_t}{C_0 + \beta \bar{C}_t} \right),$$

for which the solution can be approximated with a discrete stochastic programming method (for two assets, it is $O(n^3)$, but it is exponential in the number of assets).

Distributed multilevel modeling
D. Afshartous and G. Michailidis

Because of confidentiality reasons or storage limitations, you may want to estimate a multilinear model on distributed data (one database for each group) without exchanging too much data between the nodes. This is possible; it is not an approximation. The article also contains a terse refresher on multilevel models, with formulas for all the estimators you may want.

Introduction to statistical learning theory
O. Bousquet et al.

Good tutorial about the *theory* of supervised classifiers and the general *inequalities* you use to study them (these are inequalities with few assumptions: you are already familiar with the Markov and Chebychev ones, but there are many others).

No fishy arguments to sell you some magical algorithm as in some other articles on the subject.

Should risk managers rely on maximum likelihood estimation method while quantifying operational risk?
B. Ergashev (2007)

Operational risk is usually estimated with a loss distribution approach (LDA): for each risk cell C (line of business, event type, etc.), estimate the distribution of the severity of losses and the frequency of losses (exponential, log-normal, log-normal-gamma, log-logistic, Weibull, generalized Pareto, g -and- h , log-???; Poisson, binomial, negative binomial); these are then aggregated to produce the distribution of the total loss. To complicate things, the data we are given are often stripped of “small” losses (say, under USD 10,000); some risk cells are almost empty; and the model need not be specified properly.

The model can be estimated using a maximum likelihood estimator (MLE) or some kind of “distance” between distributions (between the empirical distribution of the data and that of the model) such as the **Cramer von Mises statistic**

$$\text{CvM}(F, G) = \int [G(x) - F(x)]^2 f(x) dx$$

or the **Anderson–Darling statistic**, almost identical, but with more weight on the tails

$$\text{AD}(F, G) = \int \frac{[G(x) - F(x)]^2}{F(x)[1 - F(x)]} f(x) dx$$

or a quantile-based distance

$$\text{QD}(F, G) = \sum_p [F^{-1}(p) - G^{-1}(p)]^2$$

where the sum is over a set of quantiles $p \in [0, 1]$, such as R’s `ppoints(N)` or the author’s “equally spaced” (sic)

$$p = \frac{1 - \varepsilon}{k}, \quad k \in \llbracket 1, N \rrbracket.$$

The estimation procedure should be robust to local extrema, e.g., simulated annealing (SA). The distance-based methods estimate the severity of losses distribution and the frequency of losses separately, while the MLE approach can estimate the joint model. The article finds the quantile distance estimation superior, with a log- T /Poisson model, simulated data in an ideal situation, simulated data with insufficient data in some cells, simulated data from a mixture of log-gaussians (I am not sure the data from such mixtures is bad enough).

You may also look at other metrics:

- Kolmogorov: $d_K(F, G) = \sum |F(x) - G(x)|$
- Lévy: $d_L(F, G) = \inf\{\varepsilon > 0 : \forall x F(x - \varepsilon) \leq G(x) \leq F(x + \varepsilon)\}$
- Total variation: $d_{TV}(F, G) = \frac{1}{2} \int |f(x) - g(x)| dx$
- Kullback–Leibler (aka entropy divergence)

$$K(F, G) = - \int \log(g/f) f(x) dx$$

- Hellinger distance: $H(F, G) = \sqrt{\int (\sqrt{f} - \sqrt{g})^2 dx}$
- Fisher information:

$$I(F||G) = \int \|\nabla \log f - \nabla \log g\|^2 f(x) dx.$$

Activity spectrum from waiting-time distribution
M. Politi and E. Scalas
arXiv:08013043

Waiting times in high-frequency financial data do not follow an exponential distribution, but can be modeled by a continuous mixture of exponentials

$$\psi(\tau) = \int_0^\infty g(\lambda) \lambda e^{-\lambda \tau} d\lambda.$$

Finding the **activity spectrum** $g(\lambda)$ from the density of durations ψ or the more directly accessible survival function (an inverse Laplace transform) is a numerically unstable task and can be tackled by a regularization method such as Tikhonov’s (this could be another name for *ridge regression*).

Personal recommendation via modified collaborative filtering
R.-R. Liu et al.
arXiv:0801.1333

An application of bipartite networks.

Voter models on heterogeneous networks

V. Sood et al.

arXiv:0712.4288

Two examples of dynamical systems on **heterogeneous graphs** (*i.e.*, graphs whose node degree distribution is “broad” – the results can be qualitatively different from those obtained on a lattice):

- Voter model: each node has a state (0 or 1), a randomly-chosen node imports its new state from a randomly chosen neighbour;
- Invasion model: idem, but replace “import” by “export”.

Critical comparison of several order-book models for stock-market fluctuations

F. Slanina

arXiv:0801.0631

There are many order book models but none reproduces the features of empirical data: return distribution, absolute return autocorrelation, Hurst exponent:

- Stigler: limit orders (buy, sell) arrive at random times and are executed if possible, or stay;
- BPS: buy and sell particles are introduced from both ends of the allowed price interval $[p_1, p_2]$, diffuse (follow a random walk), and annihilate when they meet.

Those models can be augmented with cancellations (evaporation); the other models mentioned (Genoa, Maslov) are more promising but not detailed.

Fluctuating epidemics on adaptive networks

L.B. Shaw and I.B. Schwartz

arXiv:0801.0606

Classical epidemic models (SIR, SIRS) can be studied on a fixed network or on a network changing to avoid contagion.

Asymptotic theory of statistics and probability

A. DasGupta (Springer, 2008)

A reference book, but a readable one (the style is lively and you can open it at any chapter, the author will not assume you have already read the previous 500 pages), covering everything you may want to know about convergence or limits in probability and statistics.

I only read the two chapters available on the publisher’s website.

Chapter 2 presents various **metrics** on the space of probability distributions (Kolmogorov, Lévy, total variation, Kullback–Leibler (*minimum distance estimation* is a synonym for KL estimation), Hellinger, etc.); reviews the various notions of convergence and the relations between them; recalls the *Poisson approximation* (*i.e.*, convergence to a Poisson distribution); explains that gaussian distributions are extremal (if you fix the variance) for the maximum entropy and the minimum Fisher information problems.

Chapter 29 explains what “the bootstrap works” means, and reminds us that it could fail (infinite variance, non-differentiable functions, support of F_θ that depends on θ , $\theta \in \partial\Theta$, tangency problems, etc.) – in that case, the m/n bootstrap may still work. For time series prefer fixed block sizes (there is an optimal block size that depends on the spectrum of the series); the bootstrap will not work for long-memory time series.

Comparison of volatility measures: a risk management perspective

C.T. Brownlees and G.M. Gallo (2008)

<http://ssrn.com/abstract=107321>

Volatility refers to any estimator that converges to the *integrated variance*

$$\lim_{\tau \rightarrow 0} E \left[\frac{1}{\tau} \int_T^{T+\tau} (r_t - \bar{r})^2 dt \right].$$

This assumes that you have an underlying *continuous* model of stock prices. Ideally, the estimator should be robust to the misspecification of this model.

Volatility estimators can be assessed by comparing them with a reference volatility, an implied volatility (VIX) and by checking the value at risk (VaR) and the expected shortfall (ES).

The authors find that the *daily range* is preferable to the realized volatility (which does not look at the overnight returns – the article lists several realized volatility estimators), which is better than a GARCH model.

Modern Multivariate Statistical Techniques

A.J. Izenman (Springer, 2008)

The sample chapter looked empty.

Statistical Design

G. Castella (2008)

Not understandable.

Multifractal analysis of Chinese stock volatilities based on partition function approach

Z.-Q. Jiang and Z.-X. Zhou

arXiv:0801.1710

The **multifractal analysis** of a positive time series (the realized volatility) can be carried out as follows:

- Put the time series into boxes of size s ;
- Sum the volatilities in each box;
- Normalize, *i.e.*, divide by the sum of all boxes;
- Compute:

$$\chi_q(s) = \sum_{\text{boxes}} (\text{normalized volatility})^q$$

$$\chi_q(s) \sim s^{\tau(q)}$$

$$\alpha = \frac{d\tau}{dq}$$

$$f(\alpha) = q\alpha - \tau(q).$$

***Effect of Asian currency crisis
on multifractal spectra***

**G. Oh et al.
arXiv:0801.1475**

The multifractal detrended fluctuation analysis (MF-DFA) of a time series (or returns) is performed as follows:

- Make sure the mean is 0;
- For each s , divide the data into boxes of size s ; remove a linear (or polynomial of degree m , or spline) trend; compute the average of the squared residuals; take the L^p average $F_p(s)$ of the value in each box;
- Fit $F_p(s) \sim s^{h(q)}$.

***Bayesian core: a practical approach
to computational bayesian statistics***

**J.M. Marin and C.P. Robert
Springer (2008)**

Interesting.

***Generalizing Swendsen-Wang
for image analysis***

A. Barbu and S.-C. Zhu

To sample from probability distributions on graphs (graph colourings), the Gibbs sampler performs poorly if the variables are linked (e.g., if they form a grid, as in image analysis). The Swendsen-Wang (SW) method paliates this by clustering the nodes at random (turn all the edges between nodes of different colours off, turn some of the remaining edges off at random, take the connected components – you have a different clustering each time) and changing the colour of whole clusters at a time. However, since there is no Metropolis-Hastings (MH) rule, it only works for the Ising-Potts model. It can be seen as a *data-augmentation* procedure: we actually sample from the joint distribution

Potts model \times Random Cluster Model

by moving, à la Gibbs, in only one marginal chain at a time.

This can be improved as follows:

- Use the data to get an a priori probability that two adjacent nodes have the same colour (say, a measure of pixel intensity similarity, if you are trying to segment an image) and use that probability when deciding which edges to turn off; this gives more meaningful clusters (if the probabilities are completely wrong, no real harm is done: the constant probability of the SW model is wrong as well);
- Allow the number of colours to increase or decrease: the new colour of the cluster can be a new one or an existing one.
- Use a Metropolis-Hastings (MH) acceptance probability.

The algorithm can be applied to *image segmentation* (identify the various elements of an image: car, sky, building, tree, lion, etc.).

Large graphs can be handled as follows:

- Multigrid clustering: only apply the algorithm to a subset of the data (e.g., a slice of a cube, a single image in a video, etc.);
- Multilevel clustering (for instance, in motion segmentation, this would give you a hierarchy of clustering: pixels, pixels moving in the same way (cheetah limbs), actual objects (cheetah)).

***Maximum spanning trees, asset graphs and
random matrix denoising in the analysis
of dynamics of financial networks***

**T. Heimo
arXiv:0806.4714**

You might want to denoise your correlation matrix using random matrix theory (RMT) before computing a minimum spanning tree (MST) (useful but notoriously unstable). This is actually a bad idea: the denoised matrices are even less stable and have lost most of the clustering information.

***Global recessions as a cascade phenomenon
with heterogeneous, interacting agents***

P. Ormerod (2008)

Which countries are in recession can be modeled on a (small world) graph: each node will go in recession the next year with probability α , will go out of recession with probability β , and will also go in recession if sufficiently many of its neighbours go in recession. As usual, the physicist's intuition that the shape of the network is irrelevant and can be assumed a lattice breaks down for small world graphs.

***How to quantify the influence of correlations
on investment diversification***

**M. Medo et al.
arXiv:0805.3397**

A definition of the **effective portfolio size** of a basket: the number of stocks, with iid returns (and therefore equal expected returns and risk) for which the minimum variance portfolio has the same risk. You can use the tangential portfolio instead.

***Evolutionary Monte Carlo methods
for clustering***

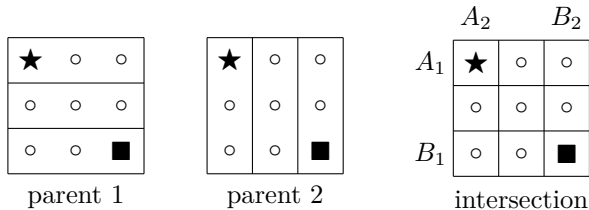
**G. Goswami et al.
Journal of computational and graphical
statistics (2007)**

Clustering algorithms either try to solve an optimization problem, more or less greedily. For instance, k -means tends to get stuck in a local optimum; you can compensate for this by running them several times and choosing the best solution; and/or you can start with a partition coming from a hierarchical clustering. MClust is similar to k -means but uses a mixture of gaussians and the BIC (bayesian information criterion) to choose the number of clusters.

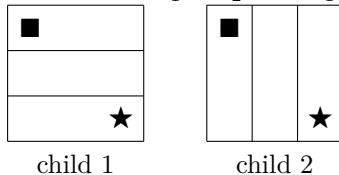
You can avoid that local optimum problem altogether by sampling from some posterior distribution.

This article suggests an improvement over MCMC sampling based on **parallel tempering** (PT) (you have several MCMC chains, each with its temperature, sampling from distributions of the form $g(z) \propto \exp(-H(z)/T)$, and we want to sample from the coldest) by adding the following operators:

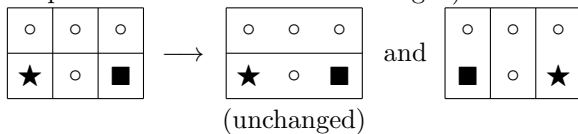
- Crossover: select two chains (parent 1 and 2), consider the intersection of the clusterings



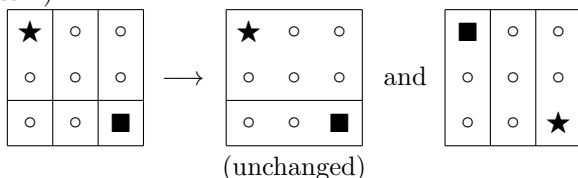
choose two clusters A and B in each parent and exchange the intersection $A_1 \cap A_2$ and $B_1 \cap B_2$



- Crossover (idem, but only select one cluster in the first parent – which remains unchanged)



- Crossover (idem, but do not reallocate the whole intersections, but only part of them (chosen at random))



- random exchange: swap two adjacent tempered chains.

In all cases, use the Metropolis-Hastings probability of the set of all tempered chains to decide whether to keep the clusterings.

This evolutionary Monte Carlo clustering (EMCC) algorithm can be used with a “Dirichlet process prior” (not defined), with an energy function $H(z)$ coming from an objective function to optimize (k -means) or a log-likelihood (mixture of gaussians, as in MClust); it can also be used for *variable selection* (just cluster the variables).

Time series decomposition and analysis in a study of oxygen isotope records M. West (1995)

The *bayesian periodogram*, *i.e.*, the posterior distribution of ω in a simple model

$$x_t = a \cos \omega t + b \sin \omega t + \varepsilon_t,$$

can help identify quasi-cyclical components. Being a whole distribution, it has a finer resolution than the classical, discrete (FFT (Fast Fourier Transform)) periodogram, which can miss frequencies between the Fourier frequencies.

The author suggests to fit an $AR(p)$ model (or an autoregressive state space model, *i.e.*, an $AR(p)$ model observed with noise) to the data, in a bayesian way: you can look at the distribution of the roots of the characteristic polynomial to test whether the data is stationary (the roots are in the $\{|z| < 1\}$ disc), you can also identify noise (roots with a small real part, *i.e.*, an argument close to $\pm\pi/2$); after somehow sorting those roots, you can look at the distribution of the arguments of the first ones: they correspond to the frequency of the *quasi-cyclical component* of the dataset (it is not extremely clear if they directly look at the roots or at the bayesian periodogram of the fitted $AR(p)$).

In a bayesian context, you can use a mixture distribution to account for or model outliers with a *contamination model*: replace $\varepsilon \sim N(0, v_0)$ by

$$\varepsilon \sim (1 - \pi)N(0, v_0) + \pi N(0, v_1)$$

with π small and $v_1 > v_0$.

Regime switching can be studied in a non-bayesian setting: cut the data in 2 or 3.

An appendix explains how to sample from (simulate) the posterior distribution of an $AR(p)$ model or a more general state space model (SSM).

Stock price jumps: news and volume play a minor role A. Joulin et al. arXiv:0803.1769

Empirical observations suggest that price jumps are not cause by news (they are followed by increased volatility while news are followed by a lower volatility) or large transaction volumes, but rather by very small-scale *liquidity squeezes* – market microstructure is a nonlinear system with feedback and has a non-mean-reverting effect.

Feasibility of portfolio optimization under coherent risk measures I. Kondor and I. Varga-Haszonits arXiv:0803.2283

The authors remark that (unconstrained) optimization of coherent risk measures (expected shortfall (ES), etc.) is sometimes unfeasible, the risk measure not being bounded below.

Shouldn't the conclusion be that constraints are necessary? The article has a more worrying shortcoming: the authors assume that the returns are elliptical (fine) and use a naive estimator of the dispersion matrix – unifying risk model estimation and optimization is good unless you botch the first step and end up with a singular matrix...

To their defense, they correctly identify the cause of the problem: if you rely on the *empirical* returns distribution, an asset will *strongly dominate* the others with non-zero probability.

***Log-normal continuous cascades:
aggregation properties and estimation.
Application to financial time series***

A. Bacry et al.
arXiv:9894.0185

The article is not understandable, but the introduction gives a definition of *multifractality* as a generalization of the “square root of time rule” to non trivial term structures:

- Square root of time: $E[X_t^2] \propto t^{1/2}$;
- Scaling behaviour: $E|X_t^q| \propto t^{\zeta q}$;
- Multifractal: $E|X_t^q| \propto t^{\zeta(q)}$, where ζ is concave but not linear.

***How much random a random network is:
a random matrix analysis***

S. Jalan and N. Bandyopadhyay
arXiv:0805.4343

To quantitatively measure the randomness of a network, one can compare the distribution of the eigenvalues of its adjacency matrix with those of a random matrix, e.g., using the Δ_3 statistic (approximately, the sum of square residuals of a linear fit of the cumulated distribution function (a staircase function) of the (empirical) distribution of the eigenvalues).

The article does not explain why we should expect the adjacency matrix of a random network to look like a random matrix.

***Bayesian approach to clustering real value,
hypergraph an bipartite graph data:
solution via variational methods***

A. Vasquez
arXiv:0805.2689

A physicist remarks that mixture models (or rather bayesian model averaging (BMA) of mixture models, since you rarely know the number of clusters) can be generalized to biclustering, *i.e.*, simultaneous clustering among both subjects and variables.

***Note on two-phase phenomena
in financial markets***

S.M. Jiang et al.
arXiv:0801:0108

Looking for a regime change (a “bifurcation”) from a unimodal to a multimodal distribution in 1-minute index returns: the authors find that this is linked to the return tails – this looks like an artefact of the way they are modeling those distributions.

***A brief history of generative models
for power law and lognormal distributions***

M. Mitzenmacher

Power law distributions, aka fat tail distributions, *i.e.*, $P[X \geq x] \sim_{x \rightarrow \infty} cx^{-\alpha}$, *i.e.*, $\log P[X \geq 0]$ is well approximated by an affine function in $\log x$ for large x , are very similar to log-normal distributions

$$\log f(x) = -\frac{(\log x)^2}{2\sigma^2} + \left(\frac{\mu}{\sigma^2} \log x\right) - \log \sqrt{2\pi}\sigma - \frac{\mu^2}{2\sigma^2}$$

(especially if σ is large) and can be used interchangeably.

Power laws can be generated by *preferential attachment*:

- Start with a 1-mode graph;
- Add a new node, link it to a uniformly-chosen node with probability α or, with probability $1 - \alpha$ to a node chosen randomly and proportionally to its degree.

This can describe the world wide web or, say, the distribution of species (degrees) among genera (nodes).

The relation between word frequency and word length can stem from an optimization problem (Huffman coding) – but simpler models, using randomness without optimization, yield similar power law distributions.

Multiplicative processes lead to log-normal distributions (e.g., Black-Scholes), but if X_t is a geometric random walk and T an exponential random variable, then X_T follows a power law.

***Topological structures
in the equities market network***

G. Leibon et al.
arXiv:0805.3470

A *multiresolution clustering* of N time series (think: stock returns) can be obtained as follows:

- Cluster the time series, using your preferred clustering algorithm (the article suggests “hierarchical spectral clustering” but assumes the reader already understands what it is);
- Compute the average time series in each cluster (think: sector returns);
- Project the time series on the subspace spanned by those average time series (this allows each time series to be considered as a member of several clusters, with weights);
- Start again with the residuals: if you are lucky, the next step can be interpreted (think: countries);
- Iterate until the eigenvalues of the correlation matrix of the residuals look like those of a random matrix, *i.e.*, until the residuals are indistinguishable from the Gaussian ensemble.

The introduction of the article contains ideas that could be developed further. For instance, random matrix theory (RMT) studies the distribution of the eigenvalues of the correlation matrix of gaussian iid variables: in statistical terms, this can be seen as the null

hypothesis of a test. One could use the same ideas with another null hypothesis: a given correlation matrix, e.g., coming from a clustering (this could be used to *test* if sectors are sufficient to explain return correlations).

Comparison of detrending methods for fluctuation analysis

A. Bashan et al.
arXiv:0805.4081

Short-range correlation in a time series is characterised by the exponential decay of the autocorrelation function

$$\rho(s) \sim \exp(-s/\tau),$$

where the decay time τ can be obtained as

$$\tau = \int_0^\infty \rho(s) ds \quad (?).$$

When this integral diverges, there are *long-term correlations*; quite often, the decay can be approximated by a power law

$$\rho(s) \sim s^{-\gamma}, \quad \gamma \in (0, 1).$$

The estimation of this exponent γ directly from the autocorrelation function ρ is very noisy: one can do better.

One can study the *scaling behaviour* (in finance, this could be called the “term structure of volatility”), *i.e.*, how the properties of the time series (e.g., the standard deviation) change with the sampling frequency (e.g., whether the economist’s “square root of time” rule is valid). More specifically:

- Somehow detrend the time series;
- Compute the *fluctuation function* $F(s)$ on scale s , *i.e.*, the standard deviation of the time series resampled at period s ;
- Try to estimate α such that

$$F(s) \sim_{s \rightarrow \infty} s^\alpha;$$

this is linked to the power law decay in the autocorrelation function by $\alpha = 1 - \gamma/2$; when $\alpha = \frac{1}{2}$, there is no long-range dependency; when $\alpha \in (\frac{1}{2}, 1)$, there is; when $\alpha \in (0, \frac{1}{2})$, there is *negative* long-range dependency; when $\alpha \geq 1$, the series is not stationary.

There are many ways of detrending the time series:

- Do not do anything: this is (almost) Hurst’s R/S ;
- Subtract the moving average on a window of size s ; always check that your chosen detrending method does not introduce bias: here, use a centered window;
- Fit a linear function on a moving window of size s and take the residuals; people often take non-overlapping windows and then worry about the discontinuities this introduces;
- Fit a polynomial of degree d on a moving window of size s and take the residuals; this is the *detrended fluctuation analysis* (DFA)

- Use a local (linear or polynomial) regression;
- Use Fourier analysis to remove known frequencies; this is limited to periodic, regular, known trends;
- Embed the time series in a higher-dimensional vector space, perform some kind of dimension reduction to identify the “trend” and take the residuals;
- Apply digital high-pass filters (Butterworth, Chebyshev, elliptical filters (?)).

Agents for traffic simulation

A. Kesting et al.
arXiv:0300

Funny article on simulating road traffic with heterogeneous agents operating in a shared environment; some of the visualization methods are... interesting – for instance, the coffee-meter to measure the driving comfort.

Efficient navigation in scale-free networks embedded in hyperbolic metric spaces

D. Krioukov et al.
arXiv:0805.1266

Routing algorithms on a graph (e.g., the internet) usually work by laborious information exchanges between neighbouring nodes – but if the graph is embedded in a metric space, if each node knows its “coordinates”, those of its neighbours and those of the target (strictly speaking, there are no coordinates in a metric space: each node only needs to know the distance between each of its neighbours and the target), one can use a *greedy* routing strategy: send packets to the neighbour closest to the destination. This could explain the *Milgram experiment* (sending letters to unknown persons, identified by name, location and job, by sending it to the friend “closest” to the target, leading to the “six degrees of separation”).

Real-world network suggest a *negatively curved* space.

However, determining the metric embedding will likely require laborious information exchanges

Accurate estimator of correlations between asynchronous signals

B. Toth and J. Kertesz
arXiv:0805.2310

Asynchronicity biases the naive correlation estimator towards zero. Just model the asynchronicity, for instance x_t and y_t could be AR(1) processes with missing values, or Ornstein-Uhlenbeck (OU) processes (not random walks: we want stationary processes) observed at discrete (different) times and try to manufacture an estimator from the cross-correlation of $\text{locf}(x)$ and $\text{locf}(y)$.

Power law in customer’s expenditures in convenience stores

T. Mizuno
arXiv:0802.0105

Yet another example of a power law distribution, in point-of-sale (POS) databases.

The structural role of weak and strong links in a financial market

A. Garas et al.
arXiv:0805.2477

Apparently, one can cluster (or apply modern/trendy tools of graph theory: scaling theory, percolation, fractal analysis, community detection (this one is not mentioned), etc.) using the correlation or their daily returns. Removing the 99% links with the largest correlation keeps the network connected.

New approach to community detection in networks

A.D. Medus and C.O. Dorso
arXiv:0808.0375

Community detection (this is the graph-theoretical word for clustering, a *community* being a subgraph “whose nodes are more connected between themselves than to external nodes”; it is often easier to look for non-overlapping clusters, *i.e.*, partitions; it can be used for “dimension” (graph size) reduction) algorithms usually try to maximize some *merit factor* such as the *modularity* of a partition of the graph

$$\sum_{\text{subgraphs}} \frac{\# \text{ external links}}{E[\# \text{ external links} | \text{degrees}]}$$

This is an NP-complete problem, mainly because the modularity is a *non-local* merit factor; furthermore, evidence suggests that modularity cannot find clusters smaller than a certain size.

There are local alternatives: a *strong community* is a subgraph C such that

$$\forall i \in C \quad k_i^{\text{in}} > k_i^{\text{out}}$$

(where k_i^{in} is the number of edges between node i and other nodes in C , and k_i^{out} is the number of edges between node i and nodes outside C); a *weak community* is a subgraph C such that

$$\sum_{i \in C} k_i^{\text{in}} > \sum_{i \in C} k_i^{\text{out}}.$$

One can find a partition into communities by maximizing

$$\sum_C \frac{\sum_{i \in C} k_i^{\text{in}} - k_i^{\text{out}}}{\sum_{i \in C} k_i}$$

under the constraints

$$\forall C \quad \forall i \in C \quad k_i^{\text{in}} > k_i^{\text{out}}$$

or

$$\forall C \quad \sum_{i \in C} k_i^{\text{in}} > \sum_{i \in C} k_i^{\text{out}}.$$

You may want to replace the strict inequalities $>$ by \leq .

Multifactor analysis of multiscaling in volatility return intervals

F. Wang et al.
arXiv:0808.3200

Many articles study the distribution of *return intervals*

$$\inf\{t : \text{Volatility}_t > q\}.$$

This one is empty.

The distribution of first-passage times and durations in FOREX and future markets

N. Sazuka et al.
arXiv:0808.0372

The distribution of the first passage time

$$T_p = \inf\{t : \text{Price}_t \geq p\}$$

is well approximated by a Weibull distribution with a power law tail.

Since inter-trade durations are not exponentially-distributed, we cannot use Lévy processes to model tick data.

A model for interevent times with long tails and multifractality in human communications: an application to financial trading

J. Perello et al.
arXiv:0805.1353

Interevent times are not Poisson; they can be modeled by mixture distributions (Poisson and something else).

Higher-order potential forces observed in bubbles and crashes in financial markets

K. Watanabe et al.
arXiv:0808.3339

Bubbles and crashes can be identified from a non-linear AR(1) model

$$y_{t+1} = f(y_t) + \text{noise}$$

where f moves from \forall through \nexists to \nexists .

Improved subset autoregression

A.I. McLeod and Y. Zhang
Journal of Statistical Software (2008)

Some time series, e.g., those with periodic features, have a complicated autoregression structure: for instance, only the first, second and eleventh partial autocorrelations are significantly non-zero (an AR(2) process with 11-year periodic patterns). You can fit this model more parsimoniously than an AR(11) with the FitAR package.

Power law distributions in empirical data

A. Clauset et al. (2007)

arXiv:0706.1062

Some interesting and clear remarks about power law estimators:

- Whether the data is continuous or discrete does matter: when using estimators for continuous data with discrete data (because you do not like zeta functions), beware of the bias they introduce;
- With fat-tailed data, least-squares-based estimators such as the classical histogram regression

$$\log p(x) = \alpha \log x + \beta,$$

should be avoided; prefer maximum likelihood estimators such as the *Hills estimator*;

- To find the cutoff beyond which your data follows a power law, you can compare the data with the corresponding quantiles (with a *Kolmogorov–Smirnov test*): if the threshold is too low, they do not match, if it is too high, they do not match either because there is too much noise; use the bootstrap if you want a confidence interval on this threshold.

Boom and bust in continuous time evolving economic model

L. Mitchell and G.J. Ackland (2008)

Economic systems (modelled here by companies competing on prices – lower prices mean higher sales but lower margins) need not converge to an equilibrium, but can exhibit *cycles*; this is similar to the Lotke–Volterra oscillations in ecology.

Heavy-tailed statistics in short-message communications

W. Hong et al. (2008)

Not all event patterns are Poisson.

Fat tails, long memory, maturity and ageing in open-source software projects

D. Challet and S. Valverde

arXiv:0802.3170

You can do a lot of interesting data analysis from a version control system (CVS, SVN, etc.) log, e.g., look at the distribution of the number of lines changed, or inter-commit time, for a given file, a given developer or the whole project; look at the cross-correlation between changeset size and intercommit time. This highlights different periods in the lifecycle of the project – if you manage programmers, you can use this idea to monitor your employees.

CVS's deficiencies (no tracking of file moves or re-names) create visible artefacts in some of the plots.

The log-periodic-AR(1)-GARCH(1,1) model for financial crashes

L. Gazola et al.

The oscillations before a crash, with slow rises and faster drops, accelerating before the singularity, can be modeled as

$$p_t = A + (t_c - t)^\beta (B + C \cos(w \log(t_c - t) + \phi)) + u_t.$$

The article suggests to use an AR(1) model with GARCH innovations for the noise term u_t .

A non-parametric investigation of risk premia

C. Peroni

Simplistic credit risk models, of the form

$$\text{corporate rate} = \text{government rate} + \text{risk premium}$$

i.e., corporate spread = risk premium are not confirmed by historical data: there are too few defaults to account for such a high risk premium. *Affine multifactor models*,

$$\text{corporate spread} \sim \text{government term structure},$$

(you can use some form of dimension reduction on the term structure: principal component analysis (PCA), factor analysis, kernel PCA, etc.) can be improved by

- allowing non-linearities (use a generalized additive model (GAM) on the (discrete) time structure);
- adding macroeconomic variables (inflation).

Stochastic volatilities and correlations, extreme values and modeling the macroeconomic environment under which Brazilian banks operate

T. Barnhill and M. Souto (IMF)

Most estimators of volatility and correlation perform equally poorly – stick to the simplest one, the exponentially weighted moving average (EWMA).

When estimating how well those estimators perform, do not compare their immediate results (volatility forecasts) but their P&L impact, as they are used in portfolio management.

Credit derivatives and risk management

M.S. Gibson (2007)

Empty article, stressing that credit derivatives do not eliminate risk; the following risk sources remain and may be tricky to quantify:

- credit risk;
- counterparty risk;
- model risk;
- rating agency risk;
- settlement risk.

Financial Modeling under non-gaussian distributions

E. Jondeau et al.

Springer Finance (2007)

This is a reference book (if you want details about any statistical *test* useful (or misleading, misused) in finance, it should be there), but not a very good one: it

emphasizes the value of the statistic of the tests rather than the p -value; the plots are often uninformative (for instance, the differences between a gaussian density and a non-gaussian density are invisible unless you use a logarithmic scale on the vertical axis) and lack axis names and units; it compares the log-likelihood of non-nested models with a different number of parameters (better use the BIC instead); etc.

Financial returns present the following *stylized facts*:

- Fat tails;
- Asymmetry (negative skew);
- Aggregated normality;
- Absence of serial correlation;
- Volatility clustering (all measures of volatility are serially correlated);
- Time-varying cross-correlation.

There are tests for all of those (but beware: some are only asymptotic; others assume that the mean and variance are known, not estimated).

Those stylized facts can be explained by the *microstructure* of financial markets (the book only considers *order-driven markets*), leading to either:

- Stable distributions (amenable to computations, through their *characteristic function* – the density can then be approximated via numeric integration);
- *Subordinated* processes, *i.e.*, the prices are geometric random walks in *market time* (aka event time or theta time), reflecting intermittent information arrival (this could be an interview question: if X_t is a brownian motion and the market time I_t has log-normal increments, what is the distribution of X_{I_t} ?)
- Mixture of distributions (estimated via the generalized method of moments (GMM), with the corresponding χ^2 test);

Current models of the *order book* (there is a detailed description of the *EKO model* of the quote arrival process) are not satisfactory.

GARCH models, and their generalizations, for instance, those based on *duration* between trades, such as ACD (autoregressive conditional duration) model (?) or CAR (compound autoregressive) model (?) can model some of the stylized facts.

GARCH models have an *aggregation* problem: if daily returns are GARCH, then weekly returns need not be GARCH; if stock returns are GARCH, then portfolio returns need not be GARCH. You can get both temporal and cross-sectional (portfolio) aggregation by enlarging the class of GARCH processes to *weakly GARCH* processes (which includes GARCH-like process with ARMA, non-gaussian innovations).

The GARCH model is over-simplified: it uses a single source of randomness, for the returns;

$$\begin{aligned}\varepsilon_t &= \sigma_t z_t \\ \sigma_t^2 &= \omega + \alpha \varepsilon_{t-1}^2 + \beta \sigma_{t-1}^2;\end{aligned}$$

stochastic volatility models have another one, for the

volatility;

$$\begin{aligned}\varepsilon_t &= \sigma_t z_t \\ \sigma_t^2 &= \exp h_t \\ h_t &= \omega + \beta \sigma_{t-1}^2 + v_t;\end{aligned}$$

they can be estimated with a *Kalman filter*.

Realized volatility, *i.e.*, volatility estimated from high-frequency data, is a discrete analogue of the *quadratic variation* (which leads to some more tests).

You can also test for the presence of jumps in the data.

Time-varying volatility is not sufficient to explain fat tails: the innovations seem to be genuinely non-gaussian.

You can still use GARCH estimators that assume the innovations are gaussian, even if they are not, and plug in estimates of the first two moments; these *quasi-maximum likelihood estimators* (QMLE) are often consistent but the resulting tests have to be robustified.

Not all values of skewness and kurtosis are attainable (the (wrongly stated) condition is simple: a set of determinants have to be positive); if you want a distribution inside some amenable family, the range of possible values is even smaller.

Beyond a few ad hoc non-gaussian distributions (say, the skewed Student T distribution), you can easily get one by a series expansions around the gaussian distribution: multiply the gaussian probability distribution function ϕ by a polynomial – if you are not careful, you end up with a non-positive “density”. Common choices are the *Gram-Charlier expansion*

$$1 + \frac{a}{6}H_3(x) + \frac{b}{24}H_4(x)$$

and the *Edgeworth expansion*

$$1 + \frac{a}{6}H_3(x) + \frac{b}{24}H_4(x) + \frac{a^2}{72}H_6(x)$$

where the *Hermite polynomials* are

$$H_i(x) = (-1)^i \frac{1}{\phi(x)} \frac{\partial^i \phi}{\partial x^i}.$$

The probability distribution function (pdf) g of a skewed version of a distribution with pdf f and cumulative distribution function (cdf) F can be obtained by *hidden truncation*,

$$g(z) = 2f(z)F(\xi z).$$

The *entropy distribution* is that with the lowest entropy among the distributions with the desired moments; if you only specify mean and variance, you get a gaussian distribution, *i.e.*, $p(x) \propto \exp P(x)$ for a degree 2 polynomial P ; if you ask for the first k moments, P just becomes a polynomial of degree (at most) p . Entropy distributions tend to be multimodal.

You can build gaussianity or goodness of fit tests from the moments, the whole density function or its restriction on some intervals.

GARCH models can be generalized by considering several time-varying moments – but beware, not all possible values of those higher moments come from an actual density function (but the GARCH-like specification of the time dependency of the higher moments can still be used to *test* for their time-dependence). More generally, you can consider *autoregressive conditional density* (ARCD) models.

There are several multivariate GARCH models, such as:

- the *vech* model, where each element of the covariance matrix V_t is a linear combination of the previous squared residuals $\varepsilon_{i,t-k}\varepsilon_{j,t-k}$ and the elements of the previous covariance matrices $V_{t-\ell}$
- the *diagonal vec model* is the special case where $V_{i,j,t}$ only depends on $\varepsilon_{i,t-k}\varepsilon_{j,t-k}$ and $V_{i,j,t-\ell}$;
- the *BEKK* model ensures that V_t is positive definite (add more time lags if you want):

$$V_{t+1} = \Omega + A'\varepsilon_t\varepsilon_t'A + B'V_tB;$$

- In higher dimensions, you can use an *orthogonal GARCH* (PCA-GARCH) model, a *factor GARCH* model, or a *flexible GARCH model* (?).

It is often easier to model the correlations and the volatilities separately (most of the previous models do not guarantee that the variance matrix remains positive definite):

- The *constant conditional correlation* (CCC) model: the correlation matrix is constant, only the volatilities are time-dependent;
- The *dynamic conditional correlation* (DCC) model,

$$\begin{aligned} u_t &= \left(\frac{\varepsilon_{it}}{\sigma_{it}} \right)_{i \in \llbracket 1, n \rrbracket} \\ \text{Cor}_t &= (\text{diag } Q_t)^{-1/2} Q_t (\text{diag } Q_t)^{-1/2} \\ Q_t &= (1 - \delta_1 - \delta_2) \bar{Q} + \delta_1 u_{t-1} u_{t-1}' + \delta_2 Q_{t-1} \\ \bar{Q} &= \text{Cov } u \end{aligned}$$

- The ARMA-like *time-varying correlation* (TVC) model,

$$R_t = (1 - \theta_1 - \theta_2)R + \theta_1 R_{t-1} + \theta_2 \Psi_{t-1}$$

where Φ_t is the sample correlation matrix of the normalized residuals;

- The *general dynamic covariance* (GDC) model (which nests most of the above).

Many 1-dimensional distributions do not have a natural multivariate generalization (the gaussian, Student or more generally elliptical distributions are exceptions); there are (several) skewed generalizations of the multivariate gaussian or Student distributions.

Choosing a copula family imposes restrictions on the various measures of concordance (Kendall's tau, Spearman's rho, etc.); for instance, many popular copulas only allow for positive dependence.

Here are a few copulas:

- Gaussian;
- Student;
- Frank, *i.e.*, archimedian with

$$\phi(t) = \log \frac{e^{-\theta} - 1}{e^{-\theta t} - 1};$$

- Clayton, *i.e.*, archimedian with

$$\phi(t) = (t^{-\theta} - 1)/\theta;$$

- Gumbel, *i.e.*, archimedian with

$$\phi(t) = (-\log t)^\theta;$$

- Plackett;
- Marshall-Olkin.

Margins and copula can be estimated together or separately and you can mix parametric and non-parametric models.

You can test the goodness of fit of a copula.

You can build a GARCH-like model by replacing the variance by the parameter(s) of the family of copulas you have chosen.

With copulas, moment computations often become analytically intractable.

There are two approaches of *extreme value theory* (EVT), that lead to estimators of the same quantities: you can focus on the distribution of maxima over subsamples, or on the tail (or *peaks over thresholds* (POT)), *i.e.*, the distribution of exceedances over a threshold.

If the (rescaled) maximum of X_1, \dots, X_n converges, then it converges to a *generalized extreme value distribution* (it encompasses the Frechet (fat tails), Gumbel (thin tails) and Weibull distributions)

$$H_\xi(x) = \exp(-(1 + \xi y)^{-1/\xi})$$

(for $\xi = 0$, take the limit). The parameter ξ is the *tail index*; its inverse $1/\xi$ is the *shape index*. You can graphically estimate this parameter with a *quantile-quantile plot* and formalize this with a regression.

If the distribution of X is in the domain of attraction of the extreme value distribution H_ξ , then the extreme distribution function

$$F_u(x) - P[X - u \leq y | X > u]$$

can be approximated by the *generalized Pareto distribution*

$$G_{\xi, \psi}(x) = 1 - \left(1 + \frac{\xi}{\psi}\right)^{-1/\xi}$$

You can also plot the *mean excess function*,

$$e(u) = E[X - u | X > u].$$

The tail index can be estimated with the *Pickands estimator*, the *Hills estimator* (popular, but it assumes we are in the domain of attraction of the Frechet distribution) or the Deckers–Einhorn–de Haan (DEdH)

estimator (a variant of the Hills estimator, that corrects for this problem, and appears more stable).

There is no satisfactory way of determining “where” the tail starts for those estimations.

For (non-iid, stationary) time series (under mild conditions), H_ξ gets replaced by H_ξ^θ (for the same ξ), where $\theta \in [0, 1]$ is called the *extremal index*.

On the other hand, the subsample maxima approach is (asymptotically) unaffected by the dependence structure.

High quantiles can be estimated from the extrema or the tail approach.

Most of the results only hold for 1-dimensional variables: the EVT limit theorems (analogues of the central limit theorem) only hold in one dimension...

With several (say, two) variables, if you get the copula wrong (and you sure will), the tail distribution will be wrong. There are three ingredients to the tail:

- The tail of the margins;
- The *asymptotic dependence*, which is unrelated to the dependence of the bulk of the data; it can be measured by

$$\chi = \lim_{u \rightarrow 1} P[V > u | U > u]$$

$$\bar{\chi} = \lim_{u \rightarrow 1} \frac{2 \log P[U > u]}{\log P[U > u, V > u]} - 1$$

(and can be estimated from (1-dimensional) tail index estimators);

- The time dependency.

To plot the tail of a bivariate random variable, consider the *Frechet transformation*

$$S = -\frac{1}{\log F_X(X)}.$$

Prefer *expected shortfall* (ES) to value at risk (VaR) (not coherent, *i.e.*, does not recognize that diversification reduces risks; ignores extreme risks) or variance (symmetric, ignores extreme risks).

VaR can be computed from a “historical simulation”, with moving window artefacts (the VaR estimates jump); the expected shortfall does not erase those artefacts. EVT-based estimators do not have this problem.

The *GARCH-EVT* model separates the conditional volatility and the tail distribution (compute the GARCH residuals via quasi-maximum likelihood (QML), *i.e.*, assuming gaussian innovations (this is consistent even if they are not gaussian); estimate the tail of the residuals; filter (?) the initial data).

The book mentions *quantile regression*, purportedly to define the conditional autoregressive value at risk (CAViaR), but their presentation is confusing and further clouded by typos.

RiskMetrics estimates volatility by an exponential (autoregressive) moving average of squared returns.

Computing the VaR of a portfolio can be tricky; if you have derivatives, you should take into account the non-linearity of those instruments.

In portfolio construction, you can approximate the investor’s utility with mean, variance and higher moments; you can also add downside risk (VaR, ES) constraints.

There are several approaches to option pricing:

- Binomial lattice;
- Replicating portfolio;
- No-arbitrage: build a risk-free portfolio with the option and the underlying; since it is risk-free and there is no arbitrage, its dynamics, *i.e.*, a stochastic differential equation (SDE), are known; since we also know the dynamics (SDE) of the underlying, we get that of the option; this eventually leads to a partial differential equation (PDE) which can be solved with the *Feynmann–Kac formula* (?);
- Martingales, *i.e.*, *risk-neutral density* (this can be seen as a “change of base”, performed to simplify the computations – and indeed, some parameters, such as the investor’s preferences and the growth of the stock even disappear);
- Pricing kernel (?).

The risk-neutral density (RND) can be estimated from the option prices. This is a non-parametric approach: there is no model of the underlying prices evolution. The authors have apparently never heard of *functional data analysis* (FDA) and happily present estimates of a density function with negative values; mixtures of distributions; Edgeworth expansions (this is the second time they explain this, but it became abominously confusing); *maximum entropy* (good to estimate density functions); splines (FDA), kernel (local) regression can give acceptable results, though, but not in the tails.

The book also explains the relation between risk-neutral and objective density: the latter incorporates the investor’s preferences and the trend (?).

The last chapter presents option pricing in a stochastic volatility model, *i.e.*, with time-dependent volatility, and the volatility, modeled by a *square root process*, comes from a second source of randomness; and in jump models (Lévy processes are often amenable if you use the characteristic function).

I have not read the appendices on stochastic calculus.

***Music constraint programming* T. Anders (2007)**

Computer aided composition (CAC) is often based on stochastic processes such as Markov chains, automata, grammars (cellular automata, fractals, L-systems, etc.) or on *constraint programming* – the topic of these presentations.

Music theory studies structures inside musical pieces, such as

- melody (how notes follow one another, in time);

- harmony (how notes can be combined, at the same time, to form a “chord”, and how chords follow one another);
- counterpoint (how melodic lines fit together);
- harmonic counterpoint (counterpoint on an underlying harmonic progression)
- musical form (macrostructure)
- orchestration (choice of instruments)

Musical representation is inherently bidimensional: a musical piece can be seen as a set of melodies, played at the same time; a set of chords, played in succession; a set of measures, played in succession; etc. List-based (or tree-based) representations (e.g., Haskore) will choose one of those representations and accessing the others will be extremely clumsy. Instead, those inclusion relations can be represented by an acyclic graph: an event (note) is both in a melody and a chord, the chords and bits of melodies are in measures, melodies and measures are in the piece.

Constraint programming can be used to compose music; traditional programming concepts (abstract data types, object-oriented programming) come in handy, and even functional programming becomes useful: the constraints (rules) are functions returning a boolean value, and you may want to transform them, *i.e.*, to specify (dynamically) where to apply them.

The *constraint store* is the set of possible values of the variables, often expressed as a cartesian product of their domains. The search algorithm progressively reduces it:

- *Propagate* the constraints, while keeping the set of potential solutions a box, *i.e.*, $\mathbf{x} \in \prod [a_i, b_i]$; for instance, given $x_1 = 3$, $x_2 \in [1, 10]$ and $x_1 \geq x_2$, the search space can be reduced to $\{3\} \times [1, 3]$;
- *Distribute* the constraints: when you can no longer propagate any constraint, choose a variable, and split its domain: for instance, $x_1 \in [1, 10]$ can be decomposed into $x_1 = 1$ and $x_1 \in [2, 10]$, or $x_1 \in [1, 5]$ and $x_1 \in [6, 10]$; the algorithm then proceeds recursively, on each branch of this tree.

When distributing the constraints, you have to choose which variable to distribute (*variable ordering*)

- *First-fail principle*: start with the variables that are the most likely to fail (if the search in a branch has to fail, it had better fail early), e.g., variables with a small domain or with a lot of constraints;
- left-to-right variable ordering: earlier notes first, longer notes first – variable ordering is used in manual composition: you classically write the harmonic progression before the actual pitches, the melody before the accompaniment;

and how to distribute it (*value ordering*)

- take the first (lowest) value;
- take a value at random (for musical purposes, this is better: the result will be more “varied”);
- use some domain-specific heuristic.

Some rules might be too complicated to be evalu-

ated with the information currently available (e.g., harmonic constraints when the rhythm is still undetermined): their *context* is *inaccessible* and should be resolved as early as possible.

Constraint programming is easy to parallelize.

Should you want to play with constraint programming in general, you can have a look at Mozart (an implementation of the Oz multi-paradigm programming language) or Gecode (C++).

Check the author’s thesis for more details on Strasheela.

Concepts, techniques and models of computer programming **P. Van Roy and S. Haridi (2003)**

This book about programming languages paradigms, more recent than Finkel’s, is based on Oz (and its implementation Mozart/Oz), a *multi-paradigm* programming language – or rather, Oz was designed to illustrate all the paradigms presented in the book – it is more a pedagogical language than a real-world one. This multiplicity is apparent in the variety of “equal” operators: `==` (test), `=` (binding), `:=` (assignment), `=:` (constraint propagation). But if you want a language with more rarely-present features, such as *dataflow variables* or *constraint programming*, it might be a good choice.

A word of caution: Mozart/Oz is not portable – it only runs on 32-bit machines.

The book is based on a series of *kernel languages*, the basic features of Oz, plus the topic of the current chapter, without any form of syntactic sugar or *linguistic abstraction* (loops, etc.): these are simpler to reason with.

The book also contains some Fortran-bashing (replacing a comma by a dot created another valid program that contributed to the loss of a satellite) and some shared-state concurrency bashing (because of a race condition in its stateful concurrent software, a medical radiation machine actually killed people).

Declarative computation model

The various scoping, typing, evaluation policies are presented:

- static (or lexical) scoping (a function, or *closure*, can use variables from the environment in which it was defined – this can be used to hide implementation details, since that environment is often no longer directly accessible);
- dynamic scoping (a function can use variables from the environment from which it is called, e.g., Vision’s special (carret) variables);
- weakly typed (you cannot get type errors: the computer will silently convert everything – even if you are trying to compare, say, a real number with a list of strings);
- strongly typed (you can get type errors);

- dynamically typed (some type checks are made at runtime, you may get type errors during the execution even though the program compiled fine);
- statically typed (the type checks are made at compile time, you cannot get type errors at run-time; Alice is apparently a variant of Oz with static typing; also note that statically typed languages need not be *explicitly typed*: some let the compiler infer the type of each variable);
- *eager* (supply-driven) evaluation (execute code thoroughly, when it appears);
- *lazy* (demand-driven) evaluation (only execute code when and if it is needed – this allows infinite data structures, such as “the list of all integers”, because they will never be evaluated completely; in Oz, define lazy functions with `fun lazy` instead of `fun`); the authors suggest that languages should be eager by default and allow some functions to be lazy;
- *non-strict* evaluation may perform more operations than lazy evaluation (which is guaranteed not to do any non-needed work) in the hope that they will be needed later (Haskell is not lazy but non-strict).

Kernel language semantics: interesting discussion on how the environment store can be implemented.

Variables can only be assigned once; they are then *bound* to a value; before that, they are *unbound*, or bound to other (unbound) variables. This is the *single-assignment store*.

As a result, there is no difference between input and output variables in a procedure: a procedure just adds information (bindings) to the store; variables that were unbound and become bound can be seen as outputs, while variables that were already bound can be seen as inputs. Functions are mere syntactic sugar: `x=f(y,z)` is equivalent to `f(?x,y,z)` (the question mark is an indication for the programmer: it is ignored by the compiler).

Declarative programming techniques

A (superficial) chapter is devoted to declarative programming techniques:

- difference between recursive computation and *iterative computation* (it looks recursive, but it uses a *bounded stack*);
- state transformations, such as *accumulators*, to turn recursive computations into iterative ones – this can be seen as a functional programming design pattern, that becomes invisible when the language provides it (states or mutable variables);
- currying;
- `foldl`, `foldr`, `map`, `filter`;
- *difference lists*;
- complexity and amortized complexity, with the *banker’s method* (put “credits” aside, for each operation, use them, if available, when needed, to perform costly operations) and the *physicist’s method* (use a *potential function* to measure how good the current state is and whose differences represent the cost of the cleaning operations,

e.g., the number of operations required to rebalance a tree, or $\Theta(1)$ thereof);

- Serialization (pickling).

For instance, most of the description of a graphical user interface (GUI) should be declarative (see a later chapter).

A *functor* is a parametrized module, that depends on some “interfaces” (aka *abstract data types* or ADT – there is apparently nothing in Oz to define types: they will only be in the programmer’s mind, as C++’s traits), for which the user will have to choose concrete implementations. A *module* is an instance of a functor. More generally, a functor is a function that returns a module – no language extension is needed, but Oz provides a linguistic abstraction.

An *open program* is a program that interacts with other programs (or humans), only known at run time; its execution is therefore not deterministic.

Declarative concurrency

Because of single-assignment, Oz’s threads (similar to a shell’s background processes) do not lead to observable non-determinism or race conditions. The *causal order* is the partial order between the execution steps. Threads can be *ready* or *suspended*; a *scheduler* decides which to run, trying not to *starve* them.

If you add *exceptions* to the concurrent model, it is no longer declarative, because the state “before the exception was thrown” is execution-dependent.

Dataflow variables can be used to communicate between threads: synchronization is implicit (no locks, etc.), dependencies can be dynamic (dictated by the data, not hardcoded), they also allow computations with partial values. A dataflow variable can be seen as both stateful (it can change state, *i.e.*, become bound – but only once) and stateless (its binding is monotonic: information can be added but neither removed nor altered).

Single-assignment variables are related to *futures* (an unbound object whose contents are being computed in separate thread) and *I-structures* (arrays of single-assignment variables).

Instead, you can use *streams* (analogue to Unix pipes). They can be demand-driven (lazy), supply-driven (eager, but if you generate data faster than you can consume it, you will run out of memory) or with a bounded buffer.

This chapter details applications such as logic gates (*lift* boolean operators to streams) and *coroutines*.

Mozart/oZ has a parser generator tool, `gump`, to extend the language with linguistic abstractions.

To accommodate demand-driven concurrency, the kernel language is augmented with a *trigger store* and a `ByNeed` primitive (that asks for its argument to be computed). (There is a problem, though: in `X==Y`, if neither `X` or `Y` is bounded or needed, they both remain unbound and unneeded; this may be a deadlock.)

Declarative concurrency can be used to let the computer determine the order of the computations, in a data-dependent way – for instance, to draw a tree, you can let the computer compute the coordinates of the points, without bothering exploring the tree yourself; or to solve the *Hamming problem*, *i.e.*, to enumerate all integers of the form $2^a 3^b 5^c$, `H=1|{Merge3{Times 2 H}{Times 3 H}{Times 5 H}}`.

Logic programming generalizes this: the constraint propagation is no longer local and the constraints are no longer equalities.

Laziness can have beneficial effects on complexity: it can turn an $O(f(n))$ *average* complexity algorithm (say, a queue implementation, which is problematic for (pure) functional languages) into a *worst-case* $O(f(n))$ complexity.

There is however a mismatch between computers (optimized to modify data) and the declarative model (which does not modify data). You can overcome some of those efficiency problems with *state*, *memoization* (but this changes the signature of the function – think Haskell *monads* – it looks awkward because it is), *instrumentation* (counting function calls – this sounds like *aspect oriented programming* (AOP)). Furthermore, the stateful model is more expressive: complex algorithms are simpler – in particular, for graph algorithms – my big problem with Haskell...

Interaction with the outside/real world brings observable nondeterminism: communicating with (independent) clients; displaying a video stream, skipping frames when needed; interacting with stateful components (hardware, libraries, UI); implementing specifications, protocols expressed in terms of state.

Impedence matching suggests to mix several computation models, for instance:

- a sequential component in a concurrent model (you need a “thread-safety enhancing” layer);
- a declarative component in a stateful model;
- a centralized component in a distributed system;
- a non-security-conscious component in a security-conscious environment (you need a “protector” – sandbox, jail, chroot, etc.);
- a non-failure-safe component in a failure-safe system (you need a fault-tolerance layer).

Message passing concurrency

Message-passing concurrency can be obtained by adding *ports* to the kernel language; they generalize streams: several threads can send messages to a port, a stream knows where the data is coming from.

Message passing can be used to implement RMI (remote method invocation – this can probably be seen as a synonym of message passing) and multi-agent systems (MAS, such as lift control systems – also used for hard disk heads). The message passed can be a *continuation*, *i.e.*, a block of code or an indication of the computations to run after the message has been processed – the recipient of the message is responsible for

the next steps of the program.

This chapter also contains an introduction to Erlang/OTP (Erlang is the language, OTP is the implementation or, rather, a set of libraries) – if you need that in a mainstream language, check the POE (Perl) and Twisted (Python) frameworks.

Here are a few concurrent message-passing patterns: several ports within a single thread (but you have to do the scheduler’s job); a queue (two ports, clients can send data to it, or retrieve data, in a FIFO manner); thread and subthread termination detection.

The *non-deterministic concurrent* model (aka concurrent logic programming) is between the declarative concurrent and the message passing one: it allows you to merge two streams (if you want more, it becomes awkward and inefficient; you would expect compiler to have been developed to make this efficient, but you would be wrong).

A port can be seen as a state: it is a list of values, which can be generated and used anywhere (contrary to a stream); if you only look at the latest value available, it behaves like a (stateful) variable. *Cells* implement the same idea, without concurrency: the model remains *sequential*.

Explicit State

State can be implicit (only present in the programmer’s mind, as with accumulators) or explicit: the kernel language can be extended with *cells*, that can be written to. Thus, Oz ends up with three equality operators: `==` (equality test), `=` (mathematical equality, binding), `:=` (assignment). Also beware of the difference between equality between cells (`X==Y`) and their contents (`@X==@Y`).

The authors try to motivate the need for states by encapsulation and compositionality – unconvincingly. *Invariants* (and encapsulation) can help reason with stateful programs (or components).

Component-based programming (encapsulation): procedures (functions and blocks of code); functors (compilation units, modules); threads; object-oriented programming adds another way of composing components: inheritance (but it adds unwanted dependencies; design patterns are the art of avoiding the resulting problems).

An *abstract data type* (ADT) can be:

- open (its structure is visible and tweakable);
- secure (encapsulated; security can be obtained through unforgeable tokens provided by the compiler or the VM, or just with lexical scoping)
- declarative (immutable, but you quickly end up creating a large number of instances; those instances have to be passed around)
- stateful (single instance; it can be a state in the scope of the procedures that need it and need not be passed around)
- unbundled (the data and the operations on the data

are stored separately – the compiler and/or virtual machine (VM) can ensure that the ADT remains secure, if needed);

- bundled.

Declarative languages (Scheme, Prolog) tend to provide open, declarative, unbundled ADTs; object-oriented ones (Java, Smalltalk) tend to provide secure, stateful, bundled ADTs.

You can use state to implement *revocable capabilities* (functions you may no longer be allowed to call after some time – just redefine them to raise an exception).

The book distinguishes the following (irrelevant?) parameter-passing mechanisms:

- call by reference
- call by variable (a special case of call-by-reference, when the argument is a cell – in procedural languages, this is often called call-by-reference);
- call by value (the procedure cannot change the value of its argument);
- call by value-result (call-by-variable, when the argument is supposed to be modified to hold the result, e.g., a procedure that would increment its argument);
- call by name (the argument is a *function* that returns a reference; a procedure with such arguments is a *thunk*);
- call by need (call-by-name, but the argument is only evaluated once).

Explicit state allows the implementation of stateful (mutable, and even extensible) *collections* (arrays, dictionaries, etc.).

Contracts (or *invariants*: what should be true before and after each operation – you can use *assertions*) can be used to formally prove *partial correctness*; to show that loops are not infinite, find some integral positive quantity that decreases at each iteration.

The state chapter contains a few digressions on software design:

- do not dilute responsibilities;
- share knowledge;
- document the *interfaces*;
- start small;
- the components can communicate with procedures, coroutines, synchronously, asynchronously, via mailboxes (asynchronously with pattern matching) or a *tuple space* (a mailbox with several recipients);
- the *interfaces* should be independent of the computation model used for the implementation

and on the future of programming:

- “Components will make programming accessible to application users”; this is already the case for statistical or digital signal processing applications (R, matlab, puredata, etc.);
- Professional programmers will combine larger components – currently, they are still too complicated (Java Beans) and their interfaces are too vaguely

documented.

State does not blend well with concurrency: whenever possible, use message-passing concurrency.

Object-Oriented Programming

Inheritance allows ADTs to be built incrementally, avoiding code duplication, but spreading the implementation. (“Early on, it was believed that inheritance would solve the problem of software reuse. This has not worked out in practice.”)

In Oz, a class is a record containing a set of attribute names and a set of methods; methods have named and optional arguments; dynamic binding (virtual methods) and static binding (non-virtual method) are decided when the method is called, not when it is defined; it allows *encapsulation* and access control (private, public)

Reflection (e.g., examining, and perhaps even changing inheritance relations at runtime), through a *meta-object protocol*, can be used for debugging, customization, separation of concerns (AOP?): method wrapping (intercepting each method call, looking at and possibly modifying its arguments), serialization (creation of a *chunk* – a few chapters earlier, it was a *pickle*), cloning, etc.

You should **not** use inheritance to structure your program, but only to create types. (Will this be exemplified in the chapter on GUIs?) In particular, inheritance should not break invariants of the parent class; you should **not** violate the *substitution property*. Some people also subclass to fix (patch) parent classes, needlessly complicating the inheritance graph.

Avoid multiple inheritance when the classes inherited from have something in common.

A *mixin* is a class often used in multiple inheritance, that does not need to know anything about the classes it will be mixed with; for instance, a **Batcher** class, with a single **batch** method, that takes a list of messages (or 0-argument methods) and sends them to **self**.

Class diagrams (UML) are useful but not a panacea: they do specify the functionality (in particular, *invariants*) of the class; they do not reflect the dynamic behaviour of the application; they display the structure of the application at a rather low level.

The OOP chapter contains an introduction to Java, should you need one, and gives a design pattern example: the *composite* (just a *tree* type).

Object-based programming is object-oriented programming without inheritance: it provides encapsulation, state shared by several methods.

If you need to define several functions f_1, \dots, f_n that can operate on several types T_1, \dots, T_m , you can opt for a *type decomposition* (define a virtual class T , let T_1, \dots, T_m inherit from it, and implement all the methods; you can easily add a new type, but adding a new function requires modifying all the classes) or a *func-*

tional decomposition (within each function, check for the type of the argument and act accordingly; it is very easy to add a new function, but cumbersome to add a new type – you have to modify all the functions). S3 classes in R are a functional decomposition – I call this *method-oriented programming*.

An *active object* is a port object which is an instance of a class, *i.e.*, an instance of a class, listening for (concurrent) messages and launching a method for each received message. The *Flavius Josephus problem* illustrates active objects: $n = 20$ soldiers are standing in a circle are committing a sequential collective suicide, by killing every $k = 3$ rd (remaining) soldier – what should be your position if you want to remain alive?

Active objects can be used to implement a simple concurrent event manager.

Shared state concurrency

Coroutines can be seen as a manual scheduling of threads in a sequential environment.

Prefer the message-passing model for multi-agent programs, consider the shared-state approach for data-centric programs (programs that access and update a large, central database).

The *maximally concurrent model* uses one thread per instruction; only data dependencies affects their execution; you do not have to explicitly create threads – you might not want to use it, though.

The kernel language provides an atomic **Exchange** operation that can be used to implement concurrent ADTs (ADTs whose operations can be run simultaneously).

Shared-state concurrent programming uses various forms of locks to create atomic actions:

- *simple lock*;
- *reentrant lock* (when a thread is inside a lock, it can enter it again (e.g., by calling the same function recursively)). *monitor* (wait points, to wait before, not after, entering a lock; e.g., a thread filling a bounded buffer would wait for enough room to be available in the buffer before requesting a lock to add an element; monitors use **wait** and **notify** operations; often, methods are *guarded*, *i.e.*, they wait until some condition is satisfied before starting);
- light (ACI) transactions (abortable, atomic, not persistent) or full ACID transactions – but how to add them to a *programming* language is still an open question, you will implement them by hand or rely on some library (or database) – databases can use *optimistic concurrency control* with *two-phase locking* (first grant locks without releasing any lock; then release locks without granting any lock; in strict 2-phase locking, all the locks are released at the same time) and *deadlock avoidance* (using the *wait-for graph*).

A *Tuple space* (aka *Linda*) is a (concurrent) multiset with non-blocking pattern matching capabilities (you

can add objects to the tuple space; you can extract one object of a given “type” to the tuple space, in a blocking or non-blocking way).

Relational programming

A *procedure* is a map (in the mathematical sense: it takes inputs and returns outputs); it can be replaced by a *relation* (in mathematical terms, you are replacing the category of sets by that of correspondances): the “function” can be multi-valued (or even sometimes fail to return a value) and the input and output arguments need not be the same for each call. You can do that in the declarative model, but you would end up with a lot of unbound variables: the relational model adds a **search** operation, that tries to assign a value to all currently unbound variables, while respecting all the constraints. There is often also a **fail** statement, which indicates that the current choice is wrong (and that another one should be tried): this allows for constraints that cannot be expressed by bounding variables, such as inequalities.

The search space can be huge and *generate-and-test* programs exhaustively examine it: this kind of *non-algorithmic programming*, without fine-tuning, can be used in databases (the amount of data is huge, but the search space small) or for toy (but perhaps instructive) examples.

Encapsulated search runs the program in an environment that controls how the search tree is explored (depth first, breadth first, etc.) and how many solutions are returned (just one, all of them, on-demand generation of new solutions, etc.); it also hides multiple bindings to the rest of the application. It can be obtained by adding a **Solve** function to the model.

Propositional logic only uses \vee , \wedge , \neg (you can use those to define \Rightarrow , \Leftrightarrow) and can be used to express, prove or disprove *tautologies*. *First-order predicate logic* adds quantifiers (\forall , \exists) and allows atoms to have arguments; for instance, it can be used to express genealogical relations and questions. For performance reasons, languages based on first-order predicate logic (Prolog) restrict the form of the axioms, e.g., to *Horn clauses*

$$\forall x_1, \dots, x_k \quad a_1 \wedge \dots \wedge a_n \implies a$$

and allow the user to provide *operational knowledge* (e.g., instead of specifying what a sort algorithm should do, specify what an efficient sort algorithm should do).

Pure Prolog does not allow higher-order programming, full prolog (which adds useful constructs with no corresponding logical semantics) only partially supports it.

Logic programming can be seen as a new “control structure” that can be added to other models (but the more stateful your program, the trickier it becomes).

Relational programming was traditionally used for natural language processing (NLP) – or, more generally, to parse ambiguous languages.

Relational programming can also be used to store re-

lations in a database; a *deductive database* can deduce tuples not explicitly stored (e.g., paths in a graph).

This chapter also contains an introduction to Prolog; most Prolog applications actually solve algorithmic problems (expert systems, deductive databases, parsing – most of the program is declarative, but the programmer somehow gives algorithmic cues to the computer).

Prolog is limited to *backtracking*, i.e., depth-first search.

Relational programming still evolves:

- Pure declarative, higher-order languages (Mercury);
- Multi-paradigm languages that allow a clearer separation of the declarative and non-declarative parts of the program (Oz);
- Constraint programming, either as a language feature (Oz, SICStus Prolog) or as a separate library (GeCode, Ilog).

Graphical User Interface Programming

GUI design is usually restricted to a single computation model such as:

- imperative (Tcl/Tk);
- object-oriented (Java);
- functional (Haskell and its fudgets);
- declarative (HTML);
- graphical.

The book advocates a multi-paradigm approach, with a declarative base augmented by objects and threads (it does not look very different from what is traditionally done; except that the widget description really *is* a data structure and (can but) does not have to be built in a procedural way):

- The structure of the widgets, their types, their initial values, their behaviour after simple events (resize) are purely declarative;
- Actions (procedures run when external events occur) are procedural;
- Handlers (objects that change the interface) are also procedural.

This allows you, for instance, to completely change the GUI (say, from Gtk to QT) while it is being used.

Distributed programming

There are various types of *distributed* systems:

- Shared memory multiprocessor;
- Distributed memory multiprocessor;
- Distributed memory multiprocessor with partial failure (so far, these are HPC (high-performance computing) and/or HA (high availability) *clusters*);
- Open distributed system (with naming and security problems).

Distributed programming is more complicated than concurrent programming:

- the processes are completely separated (they cannot share state or dataflow variables; the data has to be

converted (*marshaled*/serialized/pickled and unmarshaled/unserialized/unpickled) and sent around);

- resources (disk, memory, licence of commercial software, etc.) are localized;
- networks are several orders of magnitude slower than memory;
- failures are more common;
- not all components of the network may be trustworthy;
- nodes may be added or removed at runtime.

In the *network-transparent approach* (which assumes that none of those problems occur), the objects are local to a given node or machine (that synchronizes their usage and ensures their consistency) and respond to messages, whatever their origin – this is equivalent to concurrent programming.

Streams and ports can be readily used (the two ends of a stream can be launched in different processes). Other locking mechanisms can be used (e.g., distributed mutual exclusion using token passing).

Objects can be *stationary* (they remain on the same node – this is desirable if they access some local resources, e.g., disk) or *mobile* (the data is copied around (this is a *cache*) and the right to update the state is also moved around – this can be complemented by an *invalidation* mechanism that allows several nodes to hold a copy of the object).

Dataflow objects can be shared in an asynchronous way.

Distributed binding and distributed garbage collection can be slightly trickier.

This approach can be augmented:

- Network awareness (explicit use of message-passing for performance)
- *Naming* (language entities can be referred to by an (unforgeable) reference, name or ticket – a ticket exists outside the system – in particular, you will not be able to reclaim the memory used by an object if a ticket has been issued for it – tickets therefore often have an expiry time);
- Partial failure tolerance: you can detect (permanent) process failure (*fail-silent*) and (temporary) network failure, either in a synchronous, lazy way (you only notice the component is unavailable when you try to access it) or in an asynchronous, eager way (with a heart-beat mechanism to check what is alive and what is not; you may want to either wait or raise an exception when trying to access a failed resource); your application design should allow faults to be *confined*; fault tolerance is extremely difficult with shared states: prefer message-passing concurrency;
- Active fault tolerance (through replication and failover)
- Security (resilience to malicious failures), at several levels: application, language (check the *E* programming language), language implementation (you should not be able to tamper with compiled binaries or the virtual machine (VM)), operating system, net-

work, hardware.

Constraint programming

Alternate the following two steps:

- Local deduction: combine the constraints to reduce the space of possible solutions (the *computation space*), while keeping it manageable (e.g., a box $\prod [a_i, b_i]$); to facilitate this, the constraints are completed by *propagators*, i.e., function that explain how the constraint can help reduce the computation space (Oz already provides some of them; they can be seen as a generalization of the use of Horn clauses in Prolog);
- Search: split the problem P into $P \wedge C$ and $P \wedge \neg C$, where C is a cleverly-chosen new constraint; this choice is called a *distribution strategy*; the *search strategy* is the way the resulting tree is explored (depth-first, breadth-first or some domain-specific heuristic).

ChucK: a concurrent, on-the-fly, audio programming language

G. Wang and P.R. Cook

International Computer Music Conference
(2003)

ChucK is a *strongly-timed* audio programming language, based on data flows: data and time are clearly separated. The real-time problems are the computer's job: you just code in frozen time and say "proceed to the next sample" – or beat, or user-generated event (Midi, network, etc.). It features lightweight threads ("shreds") and a massively overloaded operator \Rightarrow (assignment, dataflow, etc.). Contrary to its predecessors, it does not seem to have escaped from the 1950s (Csound, SuperCollider) and is a real programming language, not a graphical one (Pure Data).

Also check G. Coleman's tutorial.

Beware: ChucK is not portable, it only runs on 32-bit machines.

Combining analysis and synthesis in the ChucK programming language

G. Wang, R. Fiebrink, P.R. Cook (2007)

The ChucK language has been extended with analysis operations (FFT, etc.); contrary to synthesis operations, they are not automatically computed for each sample (but usually less often) and have to be called (slightly) more explicitly; consequently, a new operator $\hat{=}$ (upchuck) was introduced besides \Rightarrow (chuck).

The Audicle: A Context-Sensitive, On-the-fly Audio Programming Environment

G. Wang and P.R. Cook

International Computer Music Conference
2004

The *Audicube* is a 3-dimensional IDE for ChucK, exploiting the real-time and on-the-fly nature of the language – think Compiz/Fusion (the faces of the

cube) and VRML (inside each face), to visualize data-flow relations, time relations (and program execution), parent-child thread relations, thread or process information à la top.

Music, a mathematical offering

D. Benson (2006)

Elementary (in its prerequisites) but excellent book that dispells a lot of misconceptions about the links between music and mathematics (in spite of a couple of overly vague mathematical statements such as "if the function f is absolutely integrable, i.e., L^1 , then $f(t) \rightarrow 0$ as $|t| \rightarrow \infty$ except perhaps on a set of measure zero", the book is eminently readable).

Here are some of the topics covered.

Sine waves are considered "pure" not because of Fourier analysis (you could use other bases) but because they are solutions of differential equations of the form $y'' + ay' + by = 0$, which describe the vibration of our eardrums (hence, the sounds we hear) and most "simple" systems (string and wood instruments).

Bessel functions, i.e. the Fourier decomposition

$$\sin(\phi + z \sin \theta) = \sum_{n=-\infty}^{\infty} J_n(z) \sin(\phi + nz)$$

appear in the description of percussion and brass instruments, *FM synthesis* and planetary motion (this last point was unclear).

Whether two sounds are consonant or dissonant does not only depend on the ratio of their fundamental frequencies, but also on their *timbre* (i.e., spectrum): when the spectrum has peaks (*partials*) at integral multiples of the fundamental frequency (string and wood instruments – such partials are called *harmonics*), the common belief that simple ratios are consonant is correct; but when the partials are not integral multiples of the fundamental (percussions and brass instruments), this is no longer true – you can even tweak the spectrum so that any interval you choose be consonant or dissonant. Pure sine wave (no partials at all) are not dissonant if they are sufficiently far apart. The "consonance function" for a given spectrum can be obtained as a linear combination of that of pure sine waves (which are obtained experimentally).

You cannot define an "instantaneous spectrum" (you have to have a non-zero window size; the smaller the window the more smeared the spectrum – this is actually *Heisenberg's uncertainty principle*), but the *Hilbert transform* can give you an *instantaneous frequency*.

The harmonics of a sound can be seen as a form of periodicity in its spectrum, which could be studied by another Fourier transform: on a logarithmic scale, this is called the *cepstrum*, $\mathcal{F} \ln \mathcal{F} f$; it is used in voice recognition to separate the partials (which are then called *formants*).

The book also presents a few musical paradoxes: *Shepard scale* (a scale whose pitch seems to be always in-

creasing, ad infinitum); *virtual pitch* (the pitch we hear is that of the fundamental, even if it is not present and only some of its integral multiples are there); *combination tones*, i.e., hearing $f_1 + f_2$, $f_1 - f_2$, $2f_1 - f_2$, $f_1 - 2f_2$ when frequencies f_1 and f_2 are sounded – this is not the *beat* phenomenon (we would hear $\frac{1}{2}(f_1 + f_2)$ and $\frac{1}{2}(f_1 - f_2)$ but results from the non-linearities (quadratic and cubic terms) in our ear – probably in the neural feedback.

When you compose music, limiting yourself to a set of allowable pitches or *scale* simplifies your work. For instance, you can start with a note and add “pleasant” multiples of it, such as octaves ($2/1$) and fifths ($3/2$). The *pythagorean scale* does just this: start with a note, add a fifth, another, and so on, but each time you exceed one octave, remove one octave; stop when you get close enough from your starting point. You never reach exactly the starting frequency, because $\log 3 / \log 2$ is irrational – its increasingly accurate rational approximations (via *continued fractions*) give rise to increasingly large scales (5, 12, 41, 53, etc.).

This set of notes is already a compromise, but we often want more: we want other simple ratios to be there. We have the octave $2/1$, the fifth $3/2$, the fourth $4/3$ (this is an octave minus a fifth, so we already have it), the third $5/4$ – more generally, *harmonics* brought down to the same octave $a/2^b$. For instance, we could use the 3-note scale $1:5/4:3/2$ (often written $4:5:6$ to emphasize that we are using the 4th, 5th and 6th harmonics – this is called a *major third*). *Just intonation* is obtained by taking three major thirds (CEG (I), the previous one FAC (IV) and the next one GBD (V)), giving 7 notes – should you want 12, you can fill in the 5 remaining ones as you want.

Classical harmony tends to move from triad to triad: the *mean-tone scale* asks for exact major thirds $5:4$ and wants the note in the middle to be exactly in the middle: CDE, FGA and GAB are exactly $1 : \sqrt{5}/2 : 5/4$ (the two remaining semitones are made equal). The problem is that the farther you move on the circle of fifths, the worse the fifths become (*wolf fifth*)

The *well-tempered scale* (or irregular temperament, or circulating temperament) tries to correct this.

Equal temperament is the easiest scale: 12 equally-spaced notes. This is the most common nowadays. The number of notes comes from the continued fraction approximating $\log 3 / \log 2$. Regarding continued fractions, you might want to notice that the golden ratio $[1, 1, 1, 1, \dots]$ is as far away as possible from the rationals.

In all those scales, the octave was pivotal: but you can remove it (especially with instruments with only odd harmonics, such as open-ended wood instruments like the clarinet) and consider, for instance, 13 notes in one octave and a half (Bohlen and Pierce) or focus on the ratios $3/2$ and $4/3$ instead of $2/1$ and $3/2$ (W. Carlos).

Since the Fourier transform of a *digital* signal is periodic, it can be written as a function of $z = \exp 2\pi i \nu \Delta t$

instead of ν : this is the *z-transform*.

Sound synthesis is slightly touched upon:

- *additive synthesis*: this is how cathedral organs work;
- *AM synthesis* (amplitude modulation): the *ADSR envelope* (attack-decay-sustain-release) is actually an example of AM synthesis;
- FM synthesis (frequency modulation), whose Fourier coefficients are *Bessel functions*; the book contains a small introduction to Csound, qualified of “tedious” – I would have said “antiquated”;
- Granular synthesis (no details)
- Phase vocoder: transform the windowed discrete Fourier transform;
- Chebychev polynomials: applying the function $T_2 : x \mapsto x^2 - 1$ doubles the frequency, i.e., $T_2(\cos \nu t) = \cos 2\nu t$; the other Chebychev polynomials are defined similarly and you can combine them, e.g., to turn sine waves into square waves.

The book ends with a chapter on symmetry – or *group theory*. Musical pieces exhibit some form of symmetry, in the rhythm (temporal) or the time \times pitch space: you can recognize the seven *frieze types*. (To avoid boredom, this is often just an approximate symmetry: for instance, the same motive can be transposed a step higher, while being constrained to remain in the same scale – some intervals will subtly change.) *Change ringing* (or *campanology*) is an Oulipo-like constraint on Church (bell) music composition, involving constrained paths in the symmetric group \mathfrak{S}_n .

In spite of its qualities, the book almost looks unfinished: when you reach the end, you still want more, you have the impression that only the surface has been scratched, that only the starting points have been exposed – in particular, there is hardly anything about composition, harmony, counterpoint, sonification or stochastic processes.

Optimization methods in portfolio management and option hedging H. Pham (2007)

Very clear (and complete) introduction to *continuous-time portfolio management*. After recalling the von Neumann–Morgenstern utility theory, with a clear motivation for the absolute and relative risk aversions and details on the *Allais paradox* (expected utility cannot reflect human preferences)

$$P_{X_1} = 0.33\delta_{2500} + 0.66\delta_{2400} + 0.01\delta_0$$

$$P_{X_2} = \delta_{2400}$$

$$P_{Y_1} = 0.34\delta_{2500} + 0.66\delta_0$$

$$P_{Y_2} = 0.33\delta_{2500} + 0.67\delta_0$$

$$X_2 > X_1, \quad Y_2 > Y_1$$

the author presents, on a discrete example, the two approaches to final-utility-maximizing multi-period portfolio management:

- Dynamic programming;
- The martingale approach:
 - Find the *risk-neutral measure*, i.e., a measure \mathbf{Q} for which the stock prices are a martingale; the price of a claim H is then $x = E^{\mathbf{Q}}[H]$;
 - Find the claim H solving the optimization problem

$$\begin{aligned} &\text{Maximize } E[U(H)] \\ &\text{such that } E^{\mathbf{Q}}[H] = x \end{aligned}$$

- Find a portfolio strategy leading to this wealth H .

The continuous-time analogues are then detailed, together with a few examples:

- Optimal strategy in a 1-stock Black-Scholes world (actively-rebalanced with constant weights);
- *Superreplication cost*, i.e., minimum initial capital of a strategy whose payoff almost surely dominates a given payoff $g(X_T)$, in a Black-Scholes world with non-constant volatility (no assumption whatsoever on the volatility, beyond measurability)

$$\frac{dX_t}{X_t} = \alpha_t dW_t$$

- Optimal strategy for a diffusion model

$$\frac{dS_t}{S_t} = \mu_t dt + \sigma_t dW_t$$

- *Quantile hedging*: since superhedging

$$P[\text{Payoff} \geq \text{Desired payoff}] = 1$$

can be expensive, one might prefer *VaR (value at risk) hedging*

$$P[\text{Payoff} \geq \text{Desired payoff}] \geq 1 - \alpha;$$

conversely, one can choose the initial investment x and look for the strategy that maximizes

$$P[\text{Payoff} \geq \text{Desired payoff}].$$

Stochastic control theory for optimal investment **M.T. Castillo and G. Parrocha**

The *Cramer–Lundberg model*, in non-life insurance, is

$$\text{Surplus} = \text{Initial capital} + \text{Income} - \text{Outflows}$$

$$R(t) = r + ct - \sum_{k=1}^{N(t)} X_k$$

$$dR = cdt - dS$$

where the aggregate claim $S(t) = \sum_{k=1}^{N(t)} X_k$ is a *compounded Poisson process* (or a *Cox process* or a *general renewal process* – those notions are not defined).

In order to control (diversify) the risk, an insurance company can invest part of its assets: the articles considers that a constant amount a is invested, a proportion $b(t)$ in a risky asset, a proportion $1 - b(t)$ in a

riskless asset.

$$dR_{\text{Risk}} = cdt - dS$$

$$\frac{d\text{Price}_0}{\text{Price}_0} = \rho dt$$

$$\frac{d\text{Price}_1}{\text{Price}_1} = \mu dt + \sigma dW$$

$$dI = a(1 - b) \frac{d\text{Price}_0}{\text{Price}_0} + ab \frac{d\text{Price}_1}{\text{Price}_1}$$

$$dU = dR + dI$$

$$U(0) = u$$

$$\text{Maximize } P[\nexists t \ U(t) < 0]$$

Finding the investment process b is a stochastic control problem, which can be solved with the Hamilton–Jacobi–Bellman (HJB) equation.

(The mathematical details in the article look fishy: before knowing anything about the solution, not even its existence, they start to assume it is twice differentiable...)

Some applications and methods of large deviations in finance and insurance **H. Pham (2007)**

Clear but rather technical presentation of *large deviation theory* and its applications in finance.

Large deviation theory is concerned with the asymptotic expansion of the probability of large events: for instance, the central limit theorem states that

$$P\left[\left|\frac{1}{n} \sum X_i - \mu\right| \geq a\right] \rightarrow_{n \rightarrow \infty} 0$$

and *Cramer’s theorem* states that

$$P\left[\frac{1}{n} \sum X_i \geq a\right] \sim_{n \rightarrow \infty} e^{-n\Gamma^*(a)}$$

$$\Gamma(\theta) = \ln E[e^{\theta X}]$$

$$\Gamma^*(x) = \sup_{\theta \in \mathbf{R}} [\theta x - \Gamma(\theta)].$$

Γ is called the *cumulant generating function* (cgf); it is the logarithm of the Laplace transform; Γ^* is called the *Fenchel–Legendre transform* of Γ ; you also encounter it as the dual of a convex function in variational analysis.

The idea behind the proof is that of *importance sampling*: change the probability distribution (in a tractable way, using an *exponential family*) to make the rare events more probable,

$$\mu_{\theta}(dx) = \exp[\theta x - \Gamma(\theta)] \mu(dx).$$

Indeed, large deviation theory can be used to prove theoretical convergence results for importance sampling.

Cramer’s theorem can be generalized: the *Gärtner–Ellis theorem* relaxes the independence assumption; *Freidlin–Wentzell theory* estimates the probability that a diffusion (a solution of a stochastic differential equation) leaves some domain, when the noise is small,

given that it remains in it when there is no noise at all (so that the event is indeed rare).

More generally, *large deviation principles* (LPD) state (more formally) that

$$P[Z_\varepsilon \in dx] \sim_{\varepsilon \rightarrow 0} e^{-I(x)/\varepsilon} dx$$

for some *rate function* I . In this context, *Varadhan's theorem* generalizes the *Laplace method*

$$\lim_{n \rightarrow \infty} \frac{1}{n} \ln \int_0^1 e^{n\phi(x)} dx = \max_{x \in [0,1]} \phi(x)$$

to

$$E \left[e^{\phi(Z_\varepsilon/\varepsilon)} \right] \sim_{\varepsilon \rightarrow 0} C \exp \left[\frac{1}{\varepsilon} \sup_x [\phi(x) - I(x)] \right].$$

Applications include:

- The *Cramer–Lundberg approximation* of the ruin probability of an insurer (initial reserve, regular premiums, iid claims arriving at the jump times of a Poisson process) when the reserve is large and its importance sampling estimator;
- Generalization to insurance-finance: if there is a drift in the asset prices, the insurer should invest and follow a buy-and-hold strategy – replacing the minimum ruin probability criterion with a less conservative one would lead to more complex/dynamic optimal strategies.
- Importance sampling to price very out-of-the-money options (change the drift to make exercise more probable);
- value at risk (VaR) estimation of a credit risk portfolio, in a 1-factor gaussian copula model, when the portfolio is large or the probability of default small.
- VaR of a passive strategy in the long term, when it gets closer and closer to the index it is tracking.

Specialising simulator generators for high-performance Monte-Carlo methods
G. Keller et al.

A scientific application of the pragmatic programmer's tip number 29, "Write code that writes code" – this could also be applied to (MCMC simulations of) mixed models (WinBugs, Jags).

This approach shifts part of the compiler design problem towards the end programmer; it looks similar to lex/yacc (or flex/bison) and the use of *hardware description languages*.

The need for open source software in machine learning
S. Sonnenburg et al.
Journal of machine learning (2007)

Open source tools are mature and suitable for large-scale real-world problems; their wider use would result in greater reuseability and interoperability – this is already the case in bioinformatics.

The article contains summary tables (definition, advantages, licences) screaming to be reused in bullet-point/Powerpoint/executive summary presentations.

The authors trace free software back to Sir Isaac Newton: *If I have seen further, it is by standing on the shoulders of giants.*

Proceedings of the Linux Audio Conference 2007

A few articles about *ambisonic* recording and *wave field synthesis* (WFS), scripting languages (PySndObj) and the usual suspects (Pure Data, etc.).

High-dimensional modelling and simulation with asymmetric normal mixtures
A. Tsanakas and A. Smith (2007)

A *gaussian mixture* is a random variable of the form $X = \sqrt{H}Z$, where $Z \sim N(0,1)$ and $H > 0$ (for instance, H could follow a discrete distribution with two values σ_1^2 and σ_2^2).

In higher dimensions, this becomes $X = \sqrt{H}LX$, where $\Sigma = LL'$ is the desired dispersion matrix and $E[H] = 1$ (H is still 1-dimensional).

This can be generalized by also randomizing the mean, yielding *asymmetric gaussian mixtures*,

$$\begin{aligned} X &= \gamma^{-1}(H-1)u + \sqrt{H}LZ \\ Z &\sim N(0, \mathbf{1}) \\ H &> 0, \quad E[H] = 1, \quad \text{Var } H = \gamma^2 \\ u &\in \mathbf{R}^n, \quad u'\Sigma^{-1}u \leq 1 \\ \Sigma &\text{ positive definite.} \end{aligned}$$

In high dimension, the dispersion matrix Σ can be parametrized by a Kronecker product.

Markov chain Monte Carlo convergence diagnostics: a comparative review M.K. Cowles and B.P. Carlin
J. Amer. Statist. Assoc. 91 (1996)

Presentation and comparison of 13 convergence diagnostics – beware, all can fail.

Constructing free energy approximations and generalized belief propagation algorithms
J.S. Yedidia et al. (2004)

Surprisingly clear presentation of the belief propagation algorithm (BP) for factor graphs.

Instead of looking for the optimal probability distribution P_{opt} , the free energy approximation looks for a simpler distribution Q (say, one where all the variables are independent (this could be seen as an "average" of one-variable models), or more generally a simply connected graphical model) that minimizes the Kullback–Leibler distance $D(P_{\text{opt}}||Q)$ – the magic is that it can be computed without knowing P_{opt} .

Variational approximations between mean field theory and the junction tree algorithm
W. Wiegerinck (2000)

Another article on the same subject.

Understanding belief propagation and its generalizations
J.S. Yedidia et al. (2002)

Earlier article on generalized belief propagation.

A differential approach to inference in bayesian networks
A. Darwiche

The author associates a polynomial (in several variables) to a bayesian network so that operations in the network (parameter estimation, marginal probability computations, etc.) are evaluations of the polynomial or its derivatives. The size of the polynomial is exponential, but manipulations on the parsing tree of the polynomial (which the author calls an “arithmetic circuit”) can make those evaluations can be amenable.

A logical approach to factoring belief networks
A. Darwiche

A more formal article on the subject.

An introduction to variational methods for graphical models
M.I. Jordan et al.
Machine Learning 37 (1999)

Variational methods come from the following remark: a convex function f is characterized by the affine functions above it,

$$f(x) = \text{Max}\{ \phi : \phi'x \geq f^*(\phi) \}.$$

To prove an assertion about f , it might suffice to prove it for all or one of those affine functions – in particular, this can lead to upper bounds of quantities involving f .

This idea can be applied to the likelihood of a graphical model: under mild assumptions (e.g., all the probabilities are from the exponential family), a graphical model is characterized by the set of independent models (graphical models with a discrete, edge-less graph – these are called *mean-field* models) more (or less) likely than it.

In particular, a graphical model can be fit by finding an independent model close to the optimal model – this is different from finding the independent model best fitting the data.

The fit can be improved by allowing more structure on the approximate graphical models: for instance, you can progressively complexify it, towards the graphical model being studied, until it becomes untractable.

Model-independent mean field theory as a local method for approximate propagation of information
M. Haft et al. (1997)

Another presentation of *mean field theory* (MFT), that suggests to average several mean field solutions (this really sounds like bayesian model averaging (BMA): for instance, you could approximate a grid by horizontal chains and vertical chains and hope to get a reasonable result by averaging those two simply connected models).

The bayesian structural EM algorithm
N. Friedman

The structure and parameters of a bayesian network can be learned “la Gibbs”, with an EM (expectation-maximization) algorithm: find the best structure for the current parameters, find the best parameters for the current structure, iterate until convergence.

Learning bayesian network models from incomplete data using importance sampling
C. Riggelsen and A. Feelders

Data augmentation (DA) is a simulation-based EM (expectation minimization) algorithm.

Probabilistic Graphical Models
M. Hauskrecht (2005)

Lecture notes, with a very useful list of references.

Probabilistic reasoning over time in Artificial Intelligence: A Modern Approach
S. Russell and P. Norvig

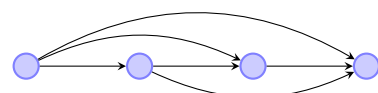
A readable introduction to dynamic bayesian networks (DBN): backward-forward message passing (for hidden Markov models (HMM)), application to the Kalman filter and extended Kalman filter (EKF), transient failure model, persistent failure model, particle filter, with applications to speech recognition (it also mentions A^* -search, with no details) and exercises.

Graphical models and automatic speech recognition
J.A. Bilmes
in Mathematical foundations of speech and language processing (2003)

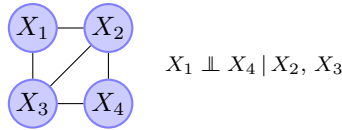
Yet another introduction to graphical models, this time targeting automatic speech recognition (ASR).

Among the examples:

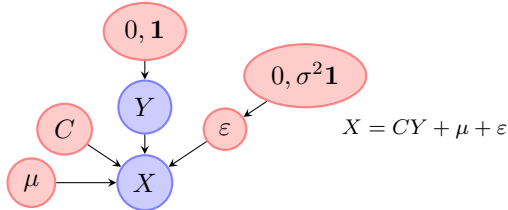
- Chain graphs (partly directed, partly undirected);
- Decision trees (yes, this is how trees look like, when turned into a bayesian network)



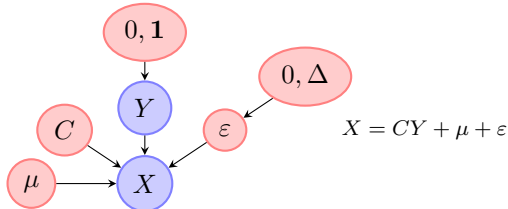
- A multivariate gaussian distribution is an undirected graphical model: the (blocks of) zeroes in the inverse of the variance matrix correspond to edges removed from the clique formed on the variables



- A multivariate gaussian distribution is a directed graphical model, the Choleski matrix yielding a “tree”, its zero entries pruning it;
- Principal component analysis (PCA):



- Factor Analysis (FA):



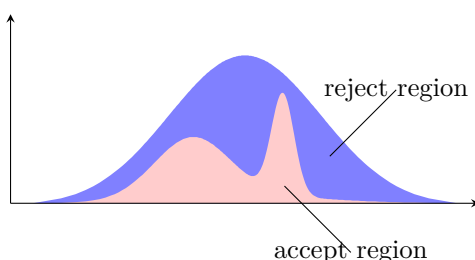
This chapter also mentions GMTK (graphical model toolkit), “open source” but with no source code available (not under active development).

Introduction to Monte Carlo methods D.J.C. MacKay

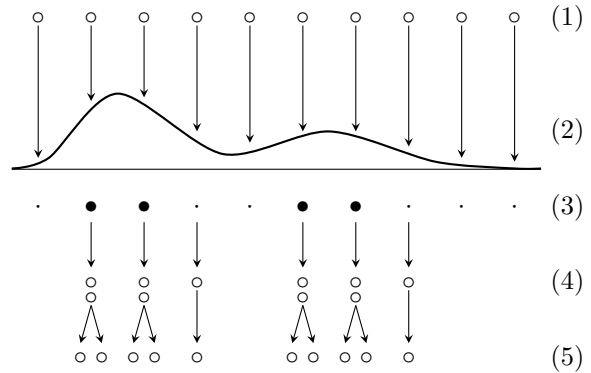
A presentation of *importance sampling*, *rejection sampling*, *Gibbs sampling* and the *Metropolis algorithm*, with some of their improvements: *hybrid Monte Carlo* (augment the state space with momentum variables), *overrelaxation* (in Gibbs sampling, do not sample from the conditional distribution, but bias it away from the current value), *simulated annealing*.

An introduction to MCMC for machine learning C. Andrieu et al. Machine Learning (2003)

Concise presentation of MCMC (Markov Chain Monte Carlo), terser than Liu’s book, with insightful pictures to represent *rejection sampling*



and *particle filters*.



Graphical models and their role in databases A. Deshpande VLDB 2007 Tutorial

A *graphical model* is a decomposition of a joint distribution on many (typically millions) variables as a product of distributions on fewer (two or three) variables.

In a *directed graphical model* or *bayesian network*, the decomposition is described by a directed acyclic graph (DAG), and the factors, called *potentials*, are conditional probabilities:

$$p(x_1, \dots, x_n) = \prod_x p(x | \text{parents}(x)).$$

In an *undirected graphical model* or *Markov random field* (MRF), the decomposition is described by the *cliques* (complete subgraphs) of a graph, and the potentials need not be probabilities (hence the need for the normalization factor Z):

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_{C \text{ Clique}} \psi_C(x_C).$$

Those directed or undirected graphs model *conditional independence* relations between sets of variables.

One can then compute marginal probabilities or the most likely labels of the remaining variables:

$$p(x_B) = \int p(x_A, x_B) dP_B$$

$$x_B | x_A = \underset{x_B}{\text{Argmax}} p(x_A, x_B).$$

Inference on a chain is easy:

$$\begin{aligned} p(x_5) &= \sum_{x_1, x_2, x_3, x_4} p(x_1, x_2, x_3, x_4) \\ &= \sum_{x_1, x_2, x_3, x_4} p(x_1) p(x_2 | x_1) p(x_3 | x_2) p(x_4 | x_3) p(x_5 | x_4) \\ &= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_2) \sum_{x_4} p(x_4 | x_3) p(x_5 | x_4) \\ &= \sum_{x_1} p(x_1) \sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_2) \underbrace{\sum_{x_4} p(x_4 | x_3) p(x_5 | x_4)}_{B_3(x_3)} \\ &\quad \underbrace{\sum_{x_2} p(x_2 | x_1) \sum_{x_3} p(x_3 | x_2) B_3(x_3)}_{B_2(x_2)} \\ &\quad \underbrace{\sum_{x_1} p(x_1) B_2(x_2)}_{B_1(x_1)} \end{aligned}$$

This can be generalized to simply-connected graphs, under the name *variable elimination algorithm*: e.g., for a hidden markov model (HMM), this is the *Viterbi algorithm*.

For more a general directed acyclic graph, just consider its *junction tree* (some articles claim this is an approximate algorithm, others (this one) that it is exact but NP-hard):

- Form the *moral graph*: join the parents of each node and forget the orientation;
- Triangulate the graph;
- The junction tree has vertices the maximal cliques of the triangulated (or *chordal*) graph and edges their separators.

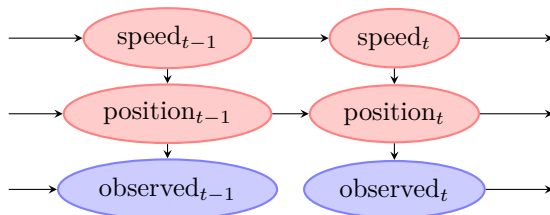
But this is NP-hard, because the cliques can be very large: you can try with approximate junction trees, whose nodes are arbitrary clusters in the triangulated graph and use *loopy belief propagation* (not detailed) if loops arise. When all else fails, MCMC (Gibbs) sampling still works.

Depending on the application, the structure of the graph can be imposed or learnt (this is NP-hard); similarly, the parameters can be imposed (from domain knowledge) or learnt.

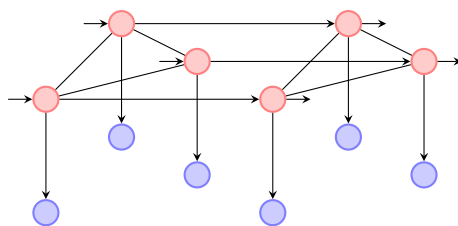
A time dimension can be added: for instance, the hidden variable model



becomes a Kalman filter



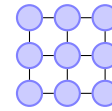
or more general *dynamic bayesian networks* (DBN) – this example is only partially oriented (this is called a *chain graph*)



General applications include:

- Error correcting codes: *turbo codes*;
- Medical diagnostic: QMR (Quick medical reference);
- Vision (Potts model or, for binary variables, Ising

model).



Database-specific applications include:

- Approximate aggregate over some of the many columns of a table (*i.e.*, marginal of the joint distribution of the columns);
- Estimate intermediate result sizes (e.g., that of `income>90 000` and `homeowner=yes`) for query plan optimization;
- Join uniformity estimation (how likely is a tuple from one relation to join with tuples from another);
- *Probabilistic relational models* (PRM, sometimes called *statistical relational models*), to extend or replace the relational model to represent uncertainties in the data;
- Record matching;
- Deduplication (with a logistic regression or a support vector machine (SVM), this can end up as a simple linear combination of the differences of the features of the records): duplication can appear in several columns and duplicates found in one can help find more in another (joint deduplication); deduplication can look for similarities in attributes (cells in a table) or records (rows).

Information extraction (e.g., named entity recognition (NER)) applications include:

- Protein and gene names from publications;
- Room attributes from hotel websites;
- Addresses and qualifications from resumes for human resources databases;
- Job postings in newsgroups;
- Emails in help centers;
- News monitoring.

Sensor networks present more challenges: the data are not only large, incomplete and imprecise, they are streaming (calling for dynamic bayesian networks), spacial (calling for added space dimensions, as we added time) and distributed – the part of the network describing the spacial distribution of the sensors can also describe the location of the databases and help devise distributed, *in-network* algorithms.

Representing and querying correlated tuples in probabilistic databases

P. Sen and A. Deshpande

Graphical models can be used to estimate tuple dependencies in a database, for query plan optimization.

Bayesian networks without tears

E. Charniak

AI Magazine (1991)

An early introductory article on bayesian networks, that allow to model probability distributions on many discrete variables without requiring “absurdly many numbers”: construct a directed acyclic graph (DAG)

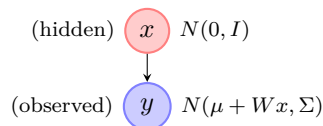
whose vertices are the variables and such that two variables x and y are conditionally independent given a set of variables Z if and only if x and y are d -separated given Z , *i.e.*, there is no d -connecting path between them – that is a path whose every interior node is either (i) linear $\bullet \rightarrow \bullet \rightarrow \bullet$ or diverging $\bullet \rightarrow \bullet \leftarrow \bullet$ and not in Z , or (ii) converging $\bullet \rightarrow \bullet \rightarrow \bullet$ and in Z or with a descendant in Z (this notion can also be explained with the *ten rules of Bayes ball*.)

The computation of every node's belief is NP-hard in general – but linear for *polytrees* (simply-connected graphs).

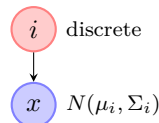
Applications include: medical diagnostic (often with some decision theory), map learning, story understanding.

An introduction to graphical models K.P. Murphy (2001)

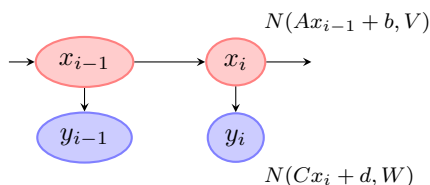
After giving a few examples of graphical models, such as factor analysis,



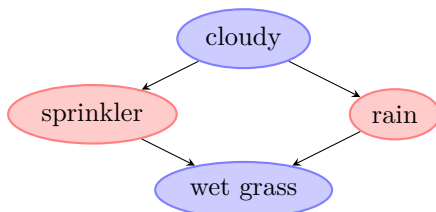
mixture of gaussians:



linear dynamic models (Kalman filters)

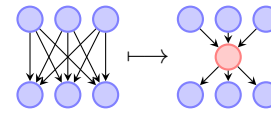


or Pearl's classical artificial intelligence example (whether the sprinkler is switched on depends on whether it is cloudy; when I come back the grass is wet: does it mean it rained?)



the article mentions, without any detail, inference algorithms, exact (variable elimination, with memoization (dynamic programming) if you need several marginals) or approximate (sampling (Monte Carlo) methods, variational methods (mean field approximation), loopy belief propagation (Pearl's algorithm, *e.g.*, used to decode *turbo codes*)).

Learning a bayesian network can be more or less complicated, depending on whether you know the graph structure (if not, you can greedily add edges) and on the presence of hidden variables – adding a hidden variable can simplify a graph:



Graphical models M.I. Jordan

Hierarchical bayesian models (as in BUGS) are a special case of graphical models.

The decomposition of the joint probability distribution

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_C f_C(x_C)$$

can also be represented with the associated *factor graph*: a bipartite graph, whose round edges are the variables and square edges the potentials f_C .

Marginal distributions can be computed (exactly) with the variable elimination algorithm: just write the humongous sum and shift the summation signs as far right as possible; this amounts to choosing an *elimination order* on the variables – choosing the best one, that will lead to the fewest operations, is NP-hard. If you need several marginals, just add memoization to the elimination algorithms – this is the *sum-product algorithm*.

One can replace sums by maximums (simply because maximum and product still form a commutative semiring) to get the most probable value of a set of variables.

Sampling algorithms can be used for approximate computations: for Gibbs sampling, you just have to condition on the *Markov blanket* of a node (its neighbours in the unoriented case; its parents, children and co-parents in the oriented case).

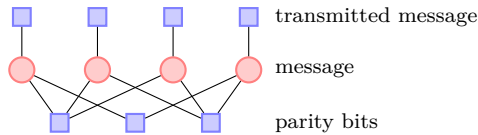
Variational inference proceeds as follows:

- Formulate the problem as an optimization problem;
- “Perturb” it, to make it simpler, typically by adding or removing constraints (inner or outer approximation)
- Solve the new problem.

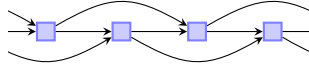
Applications include:

- Phylogenetic trees: graphical models can account for dependence between sites and lateral gene transfer (the graph structure is unknown: we can relax the requirement that it be a tree);
- Multi-locus linkage analysis in pedigrees (large pedigrees often contain many loops);
- Error correcting codes, such as *low-density parity check* (LDPC) codes, whose factor graphs look like

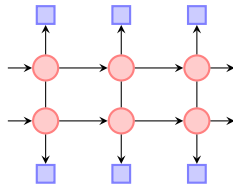
(the codes actually used are selected at random)



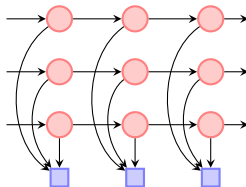
- Computational linguistics, with a host of generalizations of hidden Markov models (HMM): higher order Markov chains,



coupled HMM,



factorial HMM,



Recursive conditioning **A. Darwiche**

Cutset conditioning is an inference algorithm for graphical models that replaces the problem by several simpler ones, obtained by conditioning on (fixing the values of) a set of nodes (a cutset): this can discard loops.

Recursive conditioning applies the same idea but really cuts the graph in two at each step (you still have to choose the elimination order on the nodes). You can tune the amount of memoization used by the algorithm to control the tradeoff between its space and time complexity – it is an *any-time*, *any-space* algorithm.

Bounded RD **S. Monti and G.F. Cooper**

Another algorithm derived from *recursive decomposition* (cutset conditioning) that progressively refines interval bounds on marginal probabilities.

Inducing features of random fields **S. Della Pietra et al. (1995)**

The authors build *stepwise maxent models* as follows. Start with a graph, whose vertices are the variables to be included and whose edges represent time of spacial relations – or just take a complete graph. First add atomic features and then features of the form

atomic feature \times active feature

where the support of the candidate feature is connected in the underlying graph. To estimate the gain of a candidate feature, you can freeze the weights of the active ones: this allows the quick evaluation of thousands of features.

The article also presents the *improved iterative scaling* (IIS) algorithm.

A comparison of algorithms for maximum entropy parameter estimation **R. Malouf**

Presentation of the *improved iterative scaling* (IIS) algorithm.

Adaptive smoothing of digital images: the R package adimpro **J. Polzehl and K. Tabelow** **Journal of statistical software (2007)**

Image smoothing can be adaptive and edge-preserving: the intensity of the smoothing varies locally, with more smoothing in “homogeneous” (for some model) regions.

Interactive multivariate data analysis in R with the ade4 and ade4TkGUI packages **J. Thioulouse and S. Dray** **Journal of statistical software (2007)**

Just a GUI for the *ade4* package, that also provides factor analysis of a mixture of quantitative and qualitative variables (*dudi.hillsmith* and *dudi.mix* functions).

Numerical computing and graphics for the power method transformation using Mathematica **T.C. Headrick et al.** **Journal of statistical software (2007)**

The *power method* constructs a non-gaussian random variable Y from a gaussian one Z ,

$$Y = \sum_{k=0}^n c_k Z^k.$$

It is actually used in the other direction, to transform non-gaussian data into gaussian-looking data; the coefficients are chosen so as to match the cumulants of the data – this is a method of moments estimator.

This can be generalized to multivariate data, often as follows:

- Each Y_i is a polynomial function of a single gaussian variable Z_i , chosen to have the right cumulants;
- The Z_i are jointly gaussian, with a correlation matrix chosen to produce the second cross-moments of Y .

Efficient training of conditional random fields H. Wallach (2002)

Graphical models are a way of describing a joint distribution as a product of functions of fewer variables. For instance, with independent variables

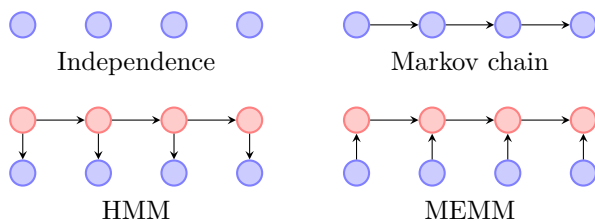
$$p(x_1, \dots, x_n) = \prod_i p(x_i);$$

or with a Markov chain

$$\begin{aligned} p(x_1, \dots, x_n) &= p(x_n | x_{n-1}, \dots, x_1) \times \\ &\quad p(x_{n-1} | x_{n-2}, \dots, x_1) \times \\ &\quad \dots \\ &\quad p(x_2 | x_1) p(x_1) \\ &= p(x_n | x_{n-1}) p(x_{n-1} | x_{n-2}) \dots \\ &\quad \dots p(x_2 | x_1) p(x_1) \\ &= p(x_1) \prod_{i>1} p(x_i | x_{i-1}). \end{aligned}$$

More generally, an oriented graph (with no cycles), defines an *oriented graphical model* (or *bayesian network*) with the decomposition

$$p(x_1, \dots, x_n) = \prod_i p(x_i | x_{\text{parents}(i)}).$$



Those models suffer from the *label bias* problem (never explained clearly).

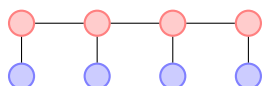
Undirected graphical models (also called *Markov random fields*) model conditional independence relations: there is one node for each variable and, given sets of variables A , B and C , A is independent of C given B iff B separates A and C , *i.e.*, iff each path from a vertex in A to a vertex in C contains a vertex from B . The *Hammersley–Clifford theorem* then says that the joint distribution can be written as

$$p(x_1, \dots, x_n) \propto \prod_{c \text{ maximal clique}} \psi_c(x_c)$$

where the factors ψ_c , called *potential functions*, need no longer be probability distributions.

Conditional random fields (CRF) are undirected graphical models for $p(y|x)$ (instead of $p(y, x)$), often with exponential potential functions

$$\psi_c(y_c | x) = \exp \sum_k \lambda_k f_k(c, y_c, x).$$



They do not exhibit the label bias problem.

The often-touted algorithms to fit maximum entropy models, iterative scaling and their generalized and improved variants (IS, GIS, IIS), that do not require the gradient to be computed, are outperformed by general optimization algorithms such as *quasi-Newton* methods (here called “variable-metric”) such as (limited memory) BFGS: indeed, the gradient and the parameters are proportional.

An introduction to conditional random field for relational learning C. Sutton and A. McCallum

Another, more detailed, review article on conditional random fields (CRF), with applications mostly in HLT/NLP (human language technology, natural language processing) names entity recognition (NER, *i.e.*, recognition of person or place names in texts), information extraction (IE, *i.e.*, automatic building of relational databases), POS (part-of-speech) tagging, shallow parsing (recognising groups of words in sentences), word sense disambiguation (WSD), etc.

The authors also model long-distance dependencies with a *skip-chain CRF*.

Conditional random fields: probabilistic models for segmenting and labeling sequence data

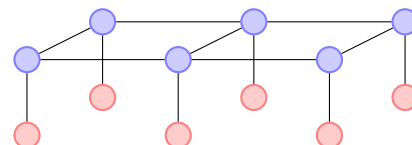
J. Lafferty et al.
Proceedings of the international conference
on machine learning (2001)

The initial article about conditional random fields (CRF) and maximum entropy Markov models (MEMM): prefer that by C. Sutton and A. McCallum, above.

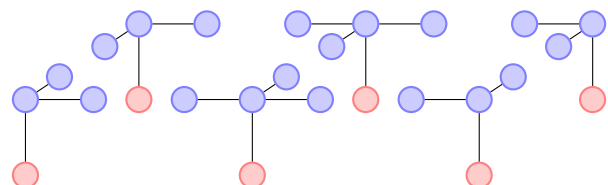
Unsupervised statistical models for general object recognition P. Carbonetto (2001)

Graphical models can also be used in image analysis, to align regions of an image (obtained by some segmentation algorithm) with human-provided annotations, and then predict those annotations.

One can neglect long-range dependencies by replacing likelihood



by pseudo-likelihood



Graphical models are usually estimated by *iterative scaling* (see somewhere below) or by heuristic algorithms such as *belief propagation*: form the *junction tree* of the graph, *i.e.*, the graph whose vertices are the maximum cliques and whose edges are their separators (it is a tree iff the graph is chordal: triangulate it first); and propagate beliefs as for a Markov chain.

The article also contains a clear (and correct – many people still think it is a data imputation algorithm) explanation of the *expectation-maximization* (EM) algorithm, to estimate a model with hidden or missing values:

- In the E-step, estimate the distribution of hidden|observed; if the hidden variables are discrete, you compute their frequency table; if the hidden variables are jointly gaussian, you compute their mean and variance; often, there is often no actual computation at this step, the distribution is just a formula that will be used in the next step;
- In the M-step, consider the joint distribution

observed, hidden|parameter,

integrate out the hidden variables using the distribution from the E-step, and maximize the corresponding expected log-likelihood; you may want to add a prior on the parameters to avoid overfitting;

- Repeat ad libidum.

A simple introduction to maximal entropy models for natural language processing **A. Ratnaparkhi (1997)**

Given a pair of discrete random variables $z = (x, y)$ and a data sample, we want to estimate the probability distribution $P(z)$ – but the random variables x and y have so many possible values (y can be the part-of-speech (pos) of a word and x the word and its context) that even with a large dataset, not all (x, y) combinations are observed.

The maximum entropy (maxent) estimator is the probability distribution with the largest entropy among those “compatible” with the data. One selects *features* $z \mapsto f_i(z)$ (for instance, boolean indicator that the word has a given prefix or is preceded by a given word) and ask that their expected value equal their observed mean. In other words, a maximum entropy estimator is an underdetermined generalized method of moments (GMM) estimator; it is the solution of a constrained optimization problem.

It can be shown that that estimator is

$$\left\{ \operatorname{Argmax}_{p \in P} H(p) \right\} = \left\{ \operatorname{Argmax}_{p \in Q} L(p) \right\} = P \cap Q$$

where

$$P = \{ p : \forall i \ E_p f_i = E_{\tilde{p}} f_i \}$$

p : model probabilities

\tilde{p} : observed probabilities

$$Q = \left\{ p : p(z) \propto \prod \alpha_i^{f_i(z)}, \alpha_i > 0 \right\}$$

$$H(p) = \sum p(z) \log p(z) \quad \text{entropy}$$

$$L(p) = \sum \tilde{p}(z) \log \tilde{p}(z) \quad \text{log-likelihood,}$$

i.e., the maximum entropy estimator is also the maximum likelihood estimator for probability distributions of the form $p(z) \propto \prod \alpha_i^{f_i(z)}$; these are sometimes called *log-linear models*, *Gibbs models*, *exponential models*, *multinomial logit models*.

This duality between maximum entropy and maximum likelihood leads to the *generalized iterative scaling* (GIS) algorithm to find the α_i . It assumes that

$$\forall z \quad \sum_i f_i(z) = C$$

$$\forall z \quad \exists i \quad f_i(z) = 1$$

$$\forall i \quad \forall z \quad f_i(z) \geq 0$$

and proceeds as follows

$$\alpha_i^{(0)} = 1$$

$$\alpha_i^{(n+1)} = \alpha_i^{(n)} \left(\frac{\tilde{E} f_i}{E^{(n)} f_i} \right)^{1/c}$$

$$E^{(n)} f_i = \sum_x p^{(n)}(z) f_i(x)$$

$$\tilde{E} f_i = \sum_x \tilde{p}(z) f_i(z) = \frac{1}{|\text{Sample}|} \sum_{x \in \text{Sample}} f_i(z).$$

While \tilde{E} is easy to compute (it is just a sum over the sample), $E^{(n)}$ usually involves an exponentially large number of terms; if $z = (x, y)$, it can be approximated as a sum over the sample $\{(x_1, y_1), \dots, (x_N, y_N)\}$:

$$E^{(n)} f_i \approx \sum_j \tilde{p}(x_j) \sum_y p^{(n)}(y|x_j) f_i(y, x_j).$$

Reduction of maximum entropy models to hidden Markov models **J. Goodman**

Maximum entropy models can be reformulated as hidden Markov models (HMM): just turn the formula in the GIS (generalized iterative scaling) algorithm into a graph (the HMM depends on the sample, though).

A gaussian prior for smoothing maximum entropy models **S.F. Chen and R. Rosenfeld (1999)**

To avoid overfitting a maximum entropy model, one can:

- Add more constraints (it is already a constrained optimization); but even reasonable constraints such as forbidding zero probabilities can make the problem unfeasible;
- Shrink the n -gram model towards the $(n - 1)$ -gram model, *i.e.*, avoid zero probabilities in $p(w_{n+1}|x_n \cdots w_1)$ by shrinking it towards $p(w_{n+1}|x_n \cdots w_2)$;
- Replace the constrained optimization

$$\text{Minimize } D(q||p_{\text{unif}}) \text{ such that } U(q) = 0$$

by a penalized entropy maximization

$$\text{Minimize } D(q||p_{\text{unif}}) + \tau U(q)$$

where $U(q)$ is often a weighted average of the squared constraint breaches

$$U(q) = \sum_i \frac{1}{\sigma_i^2} \left(\sum_x q(x) f_i(x) - \tilde{p}(x) f_i(x) \right)^2;$$

- Replace the constraints $E f_i = \tilde{E} f_i$ by fat constraints $|E f_i - \tilde{E} f_i| < \varepsilon$; this is equivalent to an entropy penalized by a step function;
- Apply the smoothing to the equivalent maximum likelihood problem: this can be interpreted as a gaussian prior.

Maximum entropy estimation in economic models with linear inequality restrictions

R.C. Campbell and R.C. Hill

You can define a maximum entropy estimator for linear regression, *i.e.*, the estimation of β in $y = x\beta + \varepsilon$, as follows: impose $\beta_i \in [\beta_i^{\min}, \beta_i^{\max}]$, and write it as $\beta_i = p_{1i}\beta_i^{\min} + p_{2i}\beta_i^{\max}$ where $p_{1i} + p_{2i} = 1$; p can be interpreted as a probability; proceed similarly for the error terms (some suggest $\varepsilon_j \in [-3\hat{\sigma}, 3\hat{\sigma}]$, where $\hat{\sigma}$ comes from some other estimator); solve the optimization problem

$$\begin{aligned} &\text{Maximize } \sum p \ln p \\ &\text{such that } y = x\beta + \varepsilon \\ &\quad p_1 + p_2 = 1. \end{aligned}$$

This can also be used for the general linear model (GLM).

Maxent models, conditional estimation and optimization

D. Klein and C. Manning
HLT-NAACL, ACL (2003)

Generative models (or joint models, such as naive Bayes classifiers, hidden Markov models (HMM), probabilistic context-free grammars, n -gram models) predict joint probabilities $p(y, x)$, while *discriminative models* (or conditional models, such as logistic regression or support vector machines (SVM) – logistic regression is actually the conditional analogue of the naive Bayes

classifier) only predict conditional probabilities $p(y|x)$. Since the latter do not have to predict the marginal distribution $p(x)$, they usually perform better.

An *exponential* (or log-linear, or maxent) model is a model of the form

$$p_\lambda(y|x) \propto \exp \sum_i \lambda_i f_i(x, y)$$

where the f_i are called *features* (often binary variables such as “the word is capitalized”, “the previous word is a verb”, “the word is ‘car’”, “the word ends in ‘-ic’”, “the word starts with ‘re-’”, “the word is in a dictionary of place names”, etc. – often, there are hundreds of thousands of them, but only a handful are non-zero).

Those models tend to overfit the data and need to be smoothed, e.g., with a prior (adding virtual data or dropping features with low counts are bad ideas – actually, including features that are always zero in the training set makes wrong answers less likely; along the same idea, redundant features, such as 2-grams and 1-grams, are an insurance against the lack of data).

This presentation also clearly presents general-purpose optimization algorithms: line search, gradient ascent (a line search in the direction of the gradient; the search path is then a succession of orthogonal segments), *conjugate gradient* (modify the gradient towards the previous search direction to avoid those right angles), Lagrange multipliers for constrained optimization.

Bivariate option pricing using dynamic copula models

R.W.J. van den Goorbergh et al.

Insurance mathematics and economics (2005)

Rainbow options, *i.e.*, options on several underlyings (call-on-max, put-on-min, call-on-min, put-on-max) can be priced by modelling the (two) underlyings as follows:

- Each underlying follows a GARCH process with gaussian innovations;
- The dependence between the innovations is described by a copula, from a family of copulas parametrized by Kendall’s τ , depending on time:

$$\tau_t = \gamma_0 + \gamma_1 \log \text{Max}(\sigma_{1,t}^2, \sigma_{2,t}^2).$$

Wouldn’t it be better to model dependence using *cointegration* instead of copulas?

One size fits all

Part 1: An idea whose time has come and gone

Part 2: Benchmarking results

M. Stonebraker et al. (2005, 2007)

Traditional database management systems (DBMS) try to provide a single solution to all your data problems; but now that some of those problems have received more attention, specialized systems outperform them by one or two orders of magnitude:

- Traditional DBMS cater to *datawarehousing* with bitmap indices (instead of B-trees), materialized views and optimizer tactic for star schema queries are outperformed by *column stores* (Vertical, MonetDB), that do not retrieve whole rows, allow for compression and better indexing or sorting; and remember that the TPC-H datawarehouse benchmark is biased to overlook problems of generalist DBMS...
- Traditional DBMS tackle *stream processing* (monitoring traffic congestion and suggesting alternative routes; tracking military vehicles and personnel; financial feeds) in a 2-stage process (update the database, then use the data); furthermore log-based recovery and synchronization needs do not fit well in a real-time setup; Stream engines (Streambase, Coral8, Apama) do not have that legacy;
- Text search (Google with GFS, BigTable and MapReduce);
- Scientific databases require specialized storage (dense or sparse matrices or higher dimensional arrays, compression (MPEG, etc.)) and optimized operations on them (matrix multiplication, pivot, interpolation, concatenation, approximate search for a small subarray); some of the authors are developing such a system, called ASAP.

***The end of an architectural era:
It's time for a complete rewrite***

**M. Stonebraker et al
VLDB 2007**

Traditional DBMS are also beaten in the OLTP arena (they authors' *H-store* prototype wins the TPC-C benchmark by almost two orders of magnitude):

- Log-based recovery goes against high availability (HA);
- Dynamic locking is outperformed by optimistic transactions (run the transaction as if there were no locking problems, unroll if and when a problem occurs – this is called transactional memory in multithreaded programming languages), especially with short-lived OLTP transactions;
- Traditional DBMS were designed at a time when data could not fit in memory: *main memory* databases can outperform them;
- The multithreading overhead is not needed, since OLTP transactions are extremely short (larger transactions should be sent to a different, OLAP system);
- Traditional DBMS are optimized for a shared memory or shared disk environment, not shared-nothing grids (with peer-to-peer communications), which provide higher reliability and hot upgrades;
- Traditional DBMS were designed in a time when hardware was expensive and people cheap and therefore require a lot of human fine-tuning: self-tuning, self-healing – self-everything – systems are eventually cheaper;
- JDBC increases the overhead and complicates or hides the program logic.

It's time to stop calling circuits "hardware"

F. Vahid (2007)

The software-hardware frontier is getting blurrier: sequential programming, event-based programming, data flow programming, *FPGA* (field programmable gate array, *i.e.*, "programmable chip") cover the whole spectrum from *temporal modelling* (formerly software) to *spacial modelling* (formerly hardware). FPGAs are still underused – some people talk of using them to speed up financial computations.

***ks: Kernel density estimation
and kernel discriminant analysis
for multivariate data in R***

**T. Duong
Journal of statistical software (2007)**

A *kernel density estimate* if

$$\hat{f}_H(x) = \frac{1}{n} \sum_i K_H(x - X_i)$$

where X_i are the observations,

$$K_H = |H|^{-1/2} K(H^{-1/2}x),$$

K is a kernel (such as $K(x) = (2\pi)^{-d/2} \exp(-\frac{1}{2}x'x)$, its choice is not crucial), and H is a positive definite matrix.

The article explains how to choose H , beyond the naive choice of "sphering the data" (replacing X by $(\text{Var } X)^{-1/2}X$ and choosing a diagonal H , check the **sm** and **KernSmooth** packages), based on the (*asymptotic*) *mean integrated square error*

$$\text{MISE}(H) = E \int_{\mathbf{R}^d} (\hat{f}_H(x) - f(x))^2 dx.$$

***Self- and super-organizing maps in R:
the kohonen package***

**R. Wehrens and L.M.C. Buydens
Journal of statistical software (2007)**

To map a high-dimensional data set to two dimensions, one can use:

- Principal component analysis (PCA);
- Multidimensional scaling (MDS): similar to PCA, but starts directly from the distance matrix, not the coordinates – there are also non-linear variants;
- self-organizing maps (SOM): a variant of the k -means algorithm that takes into account proximity between cluster centers.

SOM can also be used for supervised classification: learn two maps in parallel, one with the predictive variables, the other with the variable(s) to predict; often each will have its own distance and the algorithm will use a linear combination of these. *Super-organizing maps* generalize this to more than two layers.

Copulas: a personal view
P. Embrechts

A review article on current topics, uses, misuses, problems and workarounds in copula theory, with interesting references:

- Copulas were designed to study continuous variables (or rather: variables with continuous margins): with non-continuous variables, everything that could possibly go wrong will;
- Fitting a copula to your data is only half the story: do not forget goodness of fit tests!
- Correlation is not always a good measure: given two continuous marginal distributions F_1 and F_2 , the set of attainable correlations is a closed interval that can be much smaller than $[-1, 1]$ – e.g., with $\text{LN}(0, 1)$ and $\text{LN}(0, 4^2)$, it is $[-2.5 \cdot 10^{-4}, 1.4 \cdot 10^{-2}]$...
- There is no good answer to the question “which copula to use?”: at the very least, you should have a look at elliptic and extreme value copulas, but do not expect them to suffice – our copula catalogue is still mostly empty (some would even say useless);
- Even if you do not know which copula to choose, there is hope: given several sources of risk X_1, \dots, X_n (market risk, operational risk, etc.), the knowledge of their marginal distribution, and a risk aggregation function $\psi(X_1, \dots, X_n)$, we can find bounds of this aggregated risk

$$\psi_{\min} \leq \psi(X_1, \dots, X_n) \leq \psi_{\max}$$

For instance, $\inf\{\text{VaR}_\alpha(\sum X_i), X_i \sim F_i\}$; one sometimes also assumes properties for the copula, instead of optimizing on the set of all copulas.

A primer on copulas for count data
C. Genest and J. Nešlehová
Astin bulletin (2007)

The definition of a copula assumes that the margins are continuous: otherwise, the “copula function” $C(u, v)$ is no longer uniquely defined – and even worse, naive definitions are never a copula. Given two discrete random variables X and Y , one can consider the *set* of copulas C such that

$$\forall x, y \quad F_{X,Y}(x, y) = C(F_X(x), F_Y(y)).$$

(One of those copulas has good properties, though: the *bilinear extension*, i.e., the simplest piecewise affine, continuous copula.)

Even if the variables are independent of monotonic-dependent (this notion becomes trickier than functional monotonic dependence), this set is rather large: one can compute pointwise bounds of those copulas; these lead to bounds on Spearman’s correlation and Kendall’s τ (the difference between those bounds is at least one). The definitions of Kendall’s τ and Spearman’s ρ do not account for ties; in particular, depending on the marginal distribution, they may not span all of $[0, 1]$; they can be replaced by Kendall’s τ_b

$$\tau_b(X, Y) = \frac{\tau(X, Y)}{\sqrt{P(X_1 \neq X_2)P(Y_1 \neq Y_2)}}$$

and Spearman’s grade correlation coefficient (many other concordance measures have been suggested); dependence in the copula (positive quadrant dependence, left tail dependence, positive likelihood ratio dependence) leads to dependence in the initial variables (for the bilinear extension, this is even equivalent).

The joy of copulas:
bivariate distributions with uniform marginals
C. Genest and J. MacKay
American Statistician (1986)

Elementary (but interesting) exercises leading to the definition of archimedean copulas, an example of a singular distribution with smooth margins

$$\phi(t) = (1 - t)^\alpha, \quad \alpha \geq 1$$

$$C(x, y) = \phi^{-1}(\phi(x) + \phi(y)),$$

and a “geometric” interpretation of Kendall’s τ for archimedean copulas

$$\tau = 4 \int_0^1 \frac{\phi(t)}{\phi'(t)} dt + 1.$$

Copulas: tales and facts
T. Mikosch (2005)

Copulas do not really fit in the theory of stochastic processes and time series analysis – this is worrying, since the banking and finance industry, responsible for the *copula epidemic*, mainly use them in this context.

More work needs to be done on copula estimators and goodness of fit tests (Should we estimate margins and copula simultaneously? Should we use a parametric fit of the margins or their empirical distribution to fit the copula? Does a good fit for the margins and for the copula lead to a good fit for the multivariate distribution?).

The effects of increasing dimension (on, say, the quality of copula estimators) remains largely unstudied: most authors stick to the 2-dimensional case. For stochastic processes, you would even need infinite-dimensional copulas.

Our catalogue of copulas is still mostly empty: archimedean copulas are rarely meaningful in practical situations, elliptic copulas are bad at measuring “tail dependence” (I use quotes because the notion of upper and lower tail dependence is limited to the 2-dimensional case), there is no *canonical* multivariate distribution on which to build an extreme value copula (the Gumbel copula is popular, but is far from being the only choice).

The author also disagrees with the need to transform the margins, and with the choice of a uniform distribution.

**Discussion of
“copulas: tales and facts” by T. Mikosch
C. Genest and B. Rémillard
Extremes (2006)**

Harsh criticism of the previous article.

**Copulas: tales and facts – rejoinder
T. Mikosch**

After summarizing the responses to the article (e.g., someone mentioned that pseudo-likelihood copula estimators (use the empirical distribution of the marginals to fit a parametric copula) are inefficient), the author also clarifies what he means by “dependence structure” for stochastic processes: the whole distribution, which is determined by all the finite-dimensional marginals – this includes the 1-dimensional margins and therefore leads to large misunderstandings with copula specialists.

**Everything you always wanted to know about
copula modelling but were afraid to ask
C. Genest and A.-C. Favre
Journal of hydrologic engineering (2007)**

The following graphics can help modelling with copulas:

- Scatterplot of ranks, *i.e.*, plot of the sample copula;
- The Chi-plot plots the pairs (λ_a, χ_i) where

$$\begin{aligned} H_i &= \frac{\#\{j \neq i : X_j \leq X_i, Y_j \leq Y_i\}}{n-1} \\ F_i &= \frac{\#\{j \neq i : X_j \leq X_i\}}{n-1} \\ G_i &= \frac{\#\{j \neq i : Y_j \leq Y_i\}}{n-1} \\ \chi_i &= \frac{H_i - F_i G_i}{\sqrt{F_i(1-F_i)G_i(1-G_i)}} \\ \tilde{F}_i &= F_i - \frac{1}{2} \\ \tilde{G}_i &= G_i - \frac{1}{2} \\ \lambda_i &= 4 \operatorname{sign}(\tilde{F}_i \tilde{G}_i) \operatorname{Max}(\tilde{F}_i^2, \tilde{G}_i^2); \end{aligned}$$

large values of λ are outliers, large values of χ signal non-independence;

- The *K-plot* is the quantile-quantile plot of the H_i versus the distribution coming from independent variables (on this plot, you can also draw the curve obtained with perfect positive dependence).

The article also reviews a few copula estimators:

- If you are fitting a copula from a 1-parameter family, you can choose this parameter so as to produce the observed Kendall’s τ or Spearman’s ρ (this is a method of moments estimator); this can lead to actual tests;
- Maximum pseudo-likelihood estimators modify the log-likelihood

$$\sum_i \log c_\theta(FX_i, GX_i)$$

with the empirical marginal distributions

$$\sum_i \log c_\theta \left(\frac{R_i}{n+1}, \frac{S_i}{n+1} \right)$$

where R_i and S_i are the ranks of X_i and Y_i ;

- 2-step estimators: first estimating the marginal distributions, then the copula (this is another pseudo-likelihood);
- Kernel (non-parametric) estimators.

When modelling data with copulas, people often forget to check how good the fit is: the article presents an actual test; some graphics can also help:

- Compare the plot of the sample copula and that of simulated data from the fitted copula;
- The quantile-quantile plot of

$$\hat{C} \left(\frac{R_i}{n+1}, \frac{S_i}{n+1} \right)$$

against $C_{\hat{\theta}}(U, V)$ where $(U, V) \sim C_{\hat{\theta}}$.

The authors also remark that most measures of dependence are of the form

$$\iint J(u, v) dC(u, v)$$

where J can be chosen optimally if you know the family of copulas your data comes from; $J(u, v) = uv$ gives Spearman’s ρ , $J(u, v) = 4C(u, v) - 1$ gives Kendall’s τ , $J(u, v) = \Phi^{-1}(u)\Phi^{-1}(v)$ gives the van der Waerden statistic.

Also note that, for gaussian distributions, correlation, Spearman’s ρ and Kendall’s τ are very similar.

**Correlation and dependence in risk
management: properties and pitfalls
P. Embrechts et al. (1999)**

Correlation is a misleading measure of dependence for non-elliptic distributions, such as derivative returns in finance or loss in insurance; the article lists some of those fallacies:

- Marginal distributions and correlation need not determine the joint distribution, in particular, the value at risk (VaR) of linear portfolios is not determined by the marginal distributions and the correlation;
- Given the marginal distributions, not all correlations in $[-1, 1]$ are attainable;
- The worst-case VaR (quantile) of $X + Y$ need not occur when $\rho(X, Y)$ is maximal, *i.e.*, when X and Y are comonotonic (the variance will be maximal, but not the quantiles): the best you can say is

$$\operatorname{VaR}_\alpha(X + Y) \leq \inf_{C_\ell(u, v) = \alpha} F_1^{-1}(u) + F_2^{-1}(v);$$

this is not surprising since VaR is not a subadditive risk measure; consequently, low correlation need not mean high diversification.

The article also recalls:

- the link between correlation and regression

$$\rho(X, Y) = \frac{\text{Var } Y - \min_{a,b} E[(Y - (a + bX))^2]}{\text{Var } Y}$$

$$b = \frac{\text{Cov}(X, Y)}{\text{Var } X}$$

$$a = E[Y] - bE[X].$$

- other measures of dependence: rank correlation, Schweizer and Wolff

$$\rho_S(X, Y) = 12 \iint (C(u, v) - uv) du dv$$

$$\delta_1(X, Y) = 12 \iint |C(u, v) - uv| du dv$$

$$\delta_2(X, Y) = \left(90 \iint |C(u, v) - uv|^2 du dv \right)^{1/2}$$

$$\delta_3(X, Y) = 4 \sup_{u,v} |C(u, v) - uv|$$

comonotonicity, positive quadrant dependence

$$\forall x, y \quad P[X > x, Y > y] \geq P[X > x]P[Y > y],$$

positive association

$$\forall g_1, g_2 \text{ increasing}$$

$$E[g_1(X, Y)g_2(X, Y)] \geq E[g_1(X, Y)]E[g_2(X, Y)].$$

If all correlations are either $\rho_{\min}(F_i, F_j)$ or $\rho_{\max}(F_i, F_j)$ and satisfy some compatibility conditions, then the joint distribution is uniquely determined; it is called an *extremal distribution*.

It is not known if all linear correlation matrices are rank correlation matrices, but a sufficient condition is that the matrix of the $2\sin(\pi\rho_{ij}/6)$ be a linear correlation matrix (this is then the correlation matrix of the gaussian distribution with the prescribed rank correlation matrix, which you can use in simulations).

The article does not tackle serial and cross-correlation and estimators (of correlation, rank correlation and copulas).

On Blest's measure of rank correlation
C. Genest and J.F. Plante
Canadian journal of statistics (2003)

Many measures of concordance, for a copula C , are of the form

$$\int_{[0,1]^2} f(u, v) dC(u, v).$$

For instance:

$$\rho(X, Y) = 12 \int_{[0,1]^2} uv dC(u, v) - 3$$

$$\tau(X, Y) = 4 \int_{[0,1]^2} C(u, v) dC(u, v) - 1$$

This article examines the asymptotic distribution of Blest's measure of rank correlation, an estimator of

$$\nu(X, Y) = 2 - 12 \int_{[0,1]^2} (1 - u^2)v dC(u, v)$$

and suggest to symetrize it as

$$\xi(X, Y) = \frac{\nu(X, Y) + \nu(Y, X)}{2}$$

$$= -4 + 6 \int_{[0,1]^2} uv(4 - u - v) dC(u, v).$$

Correlation in a bayesian framework

A. DasGupta et al.

Journal of multivariate analysis (2000)

Asymptotic distribution of $\text{Cor}(\theta, \hat{\theta})$, where θ is a parameter in a model (seen as a random variable) and $\hat{\theta}$ an estimator of θ .

A characterization of quasi-copulas

C. Genest et al.

Journal of multivariate analysis (1999)

A *quasi-copula* is a function $Q : [0, 1]^2 \rightarrow [0, 1]$ so that for each track B (a path from $(0, 0)$ to $(1, 1)$, non-decreasing in both coordinates) there exist a copula C_B that coincides with Q on B .

Not all quasi-copulas are copulas.

Forecasting multifractal volatility

L. Calvet and A. Fisher (2001)

The *multifractal model of asset returns* (MMAR) is a brownian motion B compounded with a random trading time θ

$$\log P_t - \log P_0 = B[\theta(t)]$$

where the trading time θ is the cumulative distribution of a *multifractal measure*.

Given two sequences of positive real numbers m_0 and m_1 such that

$$\forall t \quad m_{0,t} + m_{1,t} = 1,$$

define a sequence of probability measures on $[0, 1]$ as follows: $\mu_0 = \lambda$ is the Lebesgue measure; μ_1 has weight $m_{0,1}$ on $[0, \frac{1}{2})$ and weight $m_{1,1}$ on $[\frac{1}{2}, 1]$; μ_2 splits each of the intervals $[0, \frac{1}{2})$ and $[\frac{1}{2}, 1]$ in two, and assigns a fraction $m_{0,2}$ (resp. $m_{1,2}$) to the left (resp. right) one; etc. In formulas,

$$\mu_n = \left(\sum_{i_1=0}^1 \cdots \sum_{i_n=0}^1 m_{i_1,1} \cdots m_{i_n,n} \mathbf{1}_{[x, x+2^{-n}]} \right) \lambda$$

where $x = \sum_{k=1}^n i_k 2^{-k}$. This is a *binomial measure* on $[0, 1]$. Instead of splitting each interval in two, one can uniformly split them into b subintervals. Instead of using deterministic sequences $m_{k,t}$, one can use a discrete random variable M taking values m_0, \dots, m_{b-1}

with probabilities p_0, \dots, p_{b-1} : this is a *random multifractal measure*. One can ask that the weight be exactly preserved at each step or only on average: $E[M] = 1/b$.

The *Poisson multifractal* also randomizes the splitting points (with a Poisson process). The authors have yet to study option pricing and stochastic portfolios in this context.

***Multifractality in asset returns:
theory and evidence***

L. Calvet and A. Fisher (1996, 2001)

Earlier article on the MMAR.

***Volatility comovement:
a multifrequency approach***

L.E. Calvet et al. (2004)

Multifractal volatility can be generalized to bivariate time series; the model can be estimated via maximum likelihood (in closed form) or a particle filter.

***Theoretical foundations of asset allocation and
pricing models with higher-order moments***

E. Jurczenko and B. Maillet (2006)

A random variable X D_4 -dominates Y if it is preferred over Y for each utility function in

$$D_4 = \{U : \mathbf{R} \rightarrow \mathbf{R} : U' > 0, U'' < 0, U''' > 0, U'''' < 0\}.$$

(This chapter was not interesting.)

***Community structure
in social and biological networks***

M. Girvan and E.J. Newman (2001)

To the usual stylized facts of social networks

- Small world: the average distance between two nodes is small, often logarithmic in the number of vertices;
- Power law degree distribution;
- Network transitivity (or clustering): the probability that two nodes are connected is higher if they are connected to a third, measured by the *clustering coefficient*

$$C = 3 \frac{\text{Number of triangles}}{\text{number of connected triples of vertices}}$$

the authors add one more: those graphs can be partitioned into communities. The following algorithm performs better than hierarchical clustering. The *betweenness* of an edge is the number of shortest paths that run through it; if there are several shortest paths between two nodes, they are given the same weight so that those weights sum up to one. Communities appear when you remove edges with the highest betweenness.

***A general approach to integrated risk
management with skewed, fat-tailed risks***

J.V. Rosenberg and Tl. Schuermann (2005)

Recipes to aggregate different sources of risk (market, credit, operational), measured by the value-at-risk (VaR), without really falling in the gaussian trap.

***Bayesian analysis for penalized spline
regression using WinBUGS***

C.M. Crainiceanu et al.

Journal of statistical software (2005)

Penalized models can be seen as bayesian models (indeed, the penalty can be turned into a prior distribution). As a result, one can tweak bayesian inference software (e.g., Bugs, Jags) into fitting penalized models.

***A software tool for the exponential power
distribution: the normalp package***

A.M. Mineo and M. Ruggieri

Journal of statistical software (2005)

General error distributions (GED), aka *exponential power distributions* (EPD), aka normal distributions of order p , are a family of distributions generalizing the gaussian distribution and designed to be used as error distributions. They have three parameters: mean, shape ($p = 2$ for the gaussian, $p = 1$ for the Laplace and $p = \infty$ for the uniform distribution) and dispersion ($\sigma_p = (E|X - \mu|^p)^{1/2}$).

Along with the usual `d`, `p`, `q`, `r` functions, the package also provides an `lmp` function for regression with EPD errors.

EbayesThresh:

R programs for empirical Bayes thresholding

I.M. Johnstone and B.W. Silverman

Journal of statistical software (2005)

When fitting data with wavelets, we want most of the coefficients to be zero – to this end, we have to threshold them. This article suggests to model the coefficients as a mixture of a Dirac density in zero (coefficients to be thresholded) and a Laplace distribution (aka *double exponential*), to which noise has been added. They use bayesian methods to perform the thresholding.

This can be applied to other situations where you have a series of random variables, measured with noise, most of which are zero.

The same idea is also used for discrete distributions, e.g., with the *zero-inflated Poisson distribution* (ZIP) – check the `zicounts` package in R.

A boosting approach for automated trading

G. Creamer and Y. Freund

Workshop on data mining for business applications, KDD 2006

Monkeying with the goodness-of-fit test

G. Marsaglia

Journal of statistical software (2005)

To test if qualitative variables x_1, \dots, x_n are iid and follow a given distribution, one can:

- Look at $Q_1 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$;
- Look at the same thing, Q_2 , for pairs $(x_1, x_2), (x_2, x_3), \dots$;
- Look at $Q_2 - Q_1$.

A clue for cluster ensembles

K. Hornik

Journal of statistical software (2005)

What to do when a clustering algorithm is not stable.

A comparison of immune and genetic algorithms for two real-life tasks of pattern recognition

A.O. Tarakanov and Y.A. Tarakanov

Journal of unconventional computing (2005)

Immunocomputing (IC), *i.e.*, k -nearest neighbours after dimension reduction via singular value decomposition (SVD), perform better than neural nets, which perform better than genetic algorithms (GA) – on those datasets, chosen because IC was working on them.

Negative selection algorithm for aircraft fault detection

A. Dasgupta et al.

Negative selection algorithms (NSA) are just a k -nearest neighbours algorithm after a principal component analysis (PCA) and/or k -means dimension reduction step.

On-line negative databases

F. Esponda et al.

Negative selection (the immune system recognizes the self by storing information about the non-self) can obscure information and help design privacy-enhancing negative databases.

Squashing flat files flatter

W. DuMouchel et al.

Instead of adapting statistical methods to large datasets, you can make large datasets smaller: random sampling is an option, but *surrogate data* can perform better: cluster the data in a large number of clusters (k -means or simply bins, with a weight on each bin), examine the statistical properties of the cluster centers, *e.g.*, moments, and generate random (or non random) data with similar properties.

The caveat is that the “statistical properties” you replicate must capture the information you want.

Downside consumption risk and expected returns

V. Polkovnichenko (2006)

Yet another article showing that investors are averse to downside risk: with *rank-dependent expected utility* (RDEU), utility is weighted by *decision weights* which are transformations of the cumulative objective probabilities of ranked events – *cumulative prospect theory* is a special case of RDEU.

Equilibrium underdiversification and the preference for skewness

T. Mitton and K. Vorkink

Review of financial studies (2007)

Preference for skewness, *i.e.*, maximization of

$$E[X] - \frac{1}{2\tau} \text{Var } X + \frac{1}{3\phi} \text{Skew } X,$$

as with “Lotto investors”, explains underdiversified portfolios.

Opinion divergence and post-earnings announcement drift

K.L. Anderson et al. (2007)

Opinion divergence can be measured as

- The *Herfindhal* measure of divergence,

$$\text{HI} = \sum_{i=1}^N \left(\frac{V_i}{V_{\text{total}}} \right)^2$$

where V_i is the volume traded by market maker i on that day;

- The standardized unexplained volume (SUV), *i.e.*, standardized residual (on the day of interest) of the regression (over the past two months)

$$\text{volume} \sim \text{returns}_+ + \text{returns}_-$$

- The standardized price volatility (SPV), *i.e.*, the standardized residuals of the regression of the daily volatility against volume and returns.

On the interaction between ultra-high frequency measures of volatility

G.M. Gallo and M. Velucchi (2007)

Comparison of realized variance $\sum x_t^2$ and *bipower variance*

$$\sum |x_t x_{t-1}|$$

(the latter is supposed to be robust to jumps) in the context of *multiplicative error models* (MEM), *i.e.*, GARCH-like models used to model positive quantities (volume, duration, range, etc.): they are almost undistinguishable.

Neoclassical factors

L. Zhang and L. Chen (2007)

The Fama–French model can be replaced by a 3-factor model: market, investment/assets, earnings/assets; this also captures *momentum* effects.

***Asset based style analysis for equity strategies:
the role of the volatility factor***

D.E. Kuenzi and X. Shi

Journal of alternative investments (2007)

When trying to explain or replicate hedge fund returns, do not forget to include a volatility factor (calls and puts (the authors' advice), change in VIX, straddle, convertible bonds), lest you remain blind to all non-linear effects, such as Lo's "capital decimation" strategy.

Who moves stock prices? Monthly evidence

B. A. Ødegaard (2007)

Change in institutional ownership might help predict monthly returns.

***A better tool than Allen's relations for
expressing temporal knowledge in interval data***

F. Morchen

**Theory and practice of temporal data mining
(2006)**

Allen's relations between intervals, used in temporal reasoning (before, meets, overlaps, starts, finished, equals), are not *robust* enough for data mining. The *time series knowledge representation* (TSKR) uses a musical analogy (tones, chords, phrases) and only considers the interaction of participating intervals and accounts for noise.

Quantile trees for marketing

C. Perlich and S. Rosset

DMBA 2006

Quantile regression can be used to focus on customers with high growth potential.

***Adaptive event detection
with time-varying Poisson processes***

A. Ihler et al.

KDD 2006

(Bugs-like) graphical model example.

Data mining in the real world

F. Fogelman Soulié

Data mining is widely used for customer relation management (CRM), fraud detection, credit scoring, etc. and *extreme data mining*, *i.e.*, data mining in databases beyond 10TB, is starting to emerge. It is characterized by:

- Heterogeneous data;
- Poor data quality (missing values, outliers – or even "unlabeled data", *i.e.*, "massively missing values");
- Large volume (billions of rows, thousands of columns), that prevents the user from hand-choosing the variables and calls for *data duplication avoidance*;
- Fast model calibration (ideally hours, should be linear in the number of rows or columns);

- Automated model quality assessment (they use a robustness indicator, KR, from Vapnik's structural risk minimization theory);
- Real-time model application;
- Industrialization (we would say "productionalization") within the same project, not as a separate, long-term project;
- Model control (check deviations on the data distribution);
- Data mining standards (JDM, PMML) to export the results;
- Process control and workflow;
- Exploratory modeling (the end-user will want to *understand* the model);
- Predictive modeling;
- Unskilled users.

***A review of some semiparametric regression
models with application to scoring***

J.L. Berthet and V. Patilea

"Scoring" means predicting a binary variable, whose values are interpreted as "bad" ("default") or "good" ("non-default"); the problem is usually asymmetric: while the "good" observations are homogeneous, there are many ways of being "bad". Between parametric methods (logistic regression, linear discriminant analysis (LDA)) and non-parametric methods (*k*-nearest neighbours (kNN), classification trees, neural networks, support vector machines), one can also consider semiparametric methods, such as logistic generalized additive model (GAM):

$$P[\text{non-default}] = m_1(X_1) + \dots + m_n(X_n)$$

$$P[\text{non-default}] = m(\lambda_1 X_1 + \dots + \lambda_n X_n)$$

This is very similar to a dimension reduction before the logistic or GAM reduction, but the dimension reduction step takes the variable to predict into account (this sounds like the difference between principal component regression (PCR) and partial least squares (PLS)).

Stochastic skew in currency options

P. Carr and L. Wu

Journal of financial economics (2007)

Traditional jump-diffusion stochastic volatility models fail to account for the time-varying skew of the risk-neutral distribution.

Time reversal invariance in finance

G. Zumbach (2007)

Most time-series models are time-reversal invariant – price time series are not...

Group dynamics of the Japanese market

W.S. Jung et al. (2007)

When building a minimum spanning tree (MST) on the correlation matrix of stock returns to find sectors, you might want to remove the influence of the global market (here, S&P 500) first.

Information flow between composite stock index and individual stocks
O. Kwon and J.-S. Yang (2007)

Individual stocks are influenced by the market.

Multiagent cooperative search for portfolio selection
D.C. Parkes and B.A. Huberman

Investment strategies can be averaged (as in bayesian model averaging (BMA) or “multi-alpha”).

Information diffusion based explanations of asset pricing anomalies
A. Bolmatis and E.G. Sekeris (2007)

For not-so-liquid stocks, you might want to replace the Fama-French factors by

- Age (number of days since the IPO);
- proportion of non-trading days in the past year.

The supervisor’s portfolio: the market price risk of German banks from 2001 to 2003 – analysis and models for risk aggregation
C. Memmel and C. Wehn
Bundesbank (2005)

The value-at-risk (VaR) of a portfolio can be estimated from the *time series* of the VaR of its constituents.

Non-linearities in exchange rate dynamics: chaos?
V. Vandrovysh (2006)

The hypothesis that exchange rates are chaotic is rejected by the *correlation dimension* and the *Lyapunov exponent*.

Shortfall: a tail of two parts
R. Martin and D. Tasche
Risk (2007)

Expected shortfall can be decomposed into a systematic and a specific part:

$$Y = X\beta + \varepsilon$$

$$E[Y|Y > y] = E[X|Y > y]\beta + E[\varepsilon|Y > y].$$

Under gaussian (or some other distributional) assumptions, this can be explicitly computed.

Cointegration and exchange market efficiency: an analysis of high frequency data
A. Trapletti et al. (1999)

Were markets efficient, a triangle of high frequency exchange rates would be cointegrated and the cointegration term would be a *martingale difference* (testing properties of the cointegration term instead of the cointegration itself is easier: it does not require a long period – just a lot of data, but with tick data, there is enough).

Markets are not efficient; arbitrage opportunities, profitable trading strategies do exist.

Predicting tick-by-tick price fluctuations by evolutionary computations
M. Tanaka-Yamawaki

Genetic algorithms (GA) can predict the direction of the next tick given previous tick directions.

The volatility effect: lower risk without lower return
D.C. Blitz and P. van Vliet (2007)

Stocks with low volatility earn higher risk-adjusted returns: equity investors overpay for risk.

Optimal hedge ratio estimation and effectiveness using ARCD
E. Kostika and R.N. Markellos

A GARCH-like model (auto-regressive conditional density) for generalized Student distributions, allowing time-varying variance, skewness and kurtosis.

Efficient time series matching by wavelets
K.P. Chan and A.W.C. Fu

The Haar wavelet transform can be used to reduce the dimension of time series, before indexing with an R-tree.

Similarity search over time series data using wavelets
I. Popivanov and R.J. Miller
ICDE (2002)

Wavelet transforms can be used as a data reduction technique for time series data – contrary to popular belief, the Haar wavelet is not always the best choice.

Time series feature extraction for data mining using DWT and DFT
F. Mörchen (2003)

When using the discrete Fourier transform (DFT, as in JPEG) or the discrete wavelet transform (DWT, as in JPEG2000) for time series dimension reduction, you can select the largest coefficients instead of the first ones.

An econometric model of serial correlation and illiquidity in hedge fund returns
M. Getmansky et al.
Journal of financial economics (2004)

Time-varying leverage and high water mark incentive fees cannot account for all the autocorrelation in hedge fund returns: illiquidity (e.g., non-synchronous trading) and/or return smoothing play a larger role.

This calls for an econometric model of smoothed returns: observed returns are a moving average of true economic (unobserved) returns. a hidden Markov chain

Mimicking portfolios with conditioning information

W. Ferson et al.

Journal of financial and quantitative analysis (2006)

To replicate a non-investible asset, you can look for the linear combination of assets in your universe whose returns have the largest correlation – this is an example of *partial least squares* (PLS).

Option pricing with aggregation of physical models and nonparametric statistical learning
J. Fan and L. Mancini (2007)

Traditional option pricing models can be combined with nonparametric (machine learning) approaches to correct their errors.

Asian options versus vanilla options
G. Ye

Asian options need not be cheaper than vanilla options.

Market timing with candlestick technical analysis
B.R. Marshall et al. (2007)

Technical analysis (TA) is not profitable....

The impact of option strategies in financial portfolios performance: mean-variance and stochastic dominance approaches
F. Abid et al. (2007)

Stochastic dominance is very rare: given to investment strategies, you should not expect one to dominate the other. However, if the strategies are related (say, a 1-parameter family of strategies, or a strategy with or without an overlay), all hope is not lost.

Individual vs aggregate preferences: the case of a small fish in a big pond
D.W. Blackburn and A.D. Ukhov (2006)

The risk preferences (utilities) of individual investors sometimes aggregate in a counterintuitive way: an economy of risk-seeking agents can be risk-averse as a whole. The notion of a “representative agent” is therefore not well defined...

Long memory modeling of realized covariance matrices
R. Chiriac and V. Voev (2007)

A time-dependent variance matrix can be modeled with an ARMA (here, a fractionally integrated ARMA) model on its Cholesky matrix.

Estimating high-frequency based (co-)variances: a unified approach
I. Nolte and V. Voev (2007)

Microstructure noise and non-synchronous observation complicates the estimation of the covariance of diffusion processes. Those effects can be studied and remedied by examining the effects of changes in the sampling frequency.

Quantile estimation with adaptive importance sampling
D. Egloff and M. Leippold (2007)

Importance sampling can also be used to estimate quantiles, not just expectations.

How well do financial and macroeconomic variables predict stock returns: time series and cross-sectional evidence
A.S. Reng Rasmussen (2006)

Yet another list of variables to predict future returns.

What happened to the quants in August 2007?
A.E. Khandani and A.W. Lo (2007)

This may have been caused by the unwind of a large portfolio.

The paradox of quantitative investing and ways to deal with it
T. Suwabe et al.
Goldman Sachs (2007)

The authors build a BMA-CAPM (*bayesian model averaging* capital asset pricing model) on a hundred variables, to decompose the momentum (in excess of the industry average) into common and specific components: the latter can be used as a reversal (it used to work), the former as a momentum (noisier, only works in the past three years).

One could also try:

- a time-series (Northfield-like) risk model (this means one regression per date and per stock);
- support vector machines (SVM) or regularization path regression (least angle regression (LAR), lasso, etc.) instead of BMA.

Costly short-selling and stock price adjustment to earnings announcements
A.V. Reed (2007)

Costly short-selling, or short-selling constraints, reduces the speed at which the price adjusts to private information, increases volatility (around earnings announcements), reduces the information content of price and volume. (This looks like a textbook example of the omitted variable syndrome...)

A stochastic volatility model with fat tails, skewness and leverage effects
D.R. Smith (2007)

One can try to reproduce the **leverage effect**, $\text{Cor}(r_t, v_{t+1}) < 0$, *i.e.*, shocks are correlated with future volatility, as follows.

- This first attempt exhibits leverage, but not negative skewness;

$$\begin{aligned} y_t &= \mu + e^{x_t/2} + \varepsilon_t \\ x_{t+1} &= \omega + \phi_t x_t + \eta_t \\ \begin{pmatrix} \varepsilon_t \\ \eta_t \end{pmatrix} &\sim N \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{pmatrix} \right) \end{aligned}$$

- If you ask that shocks be correlated with contemporaneous volatility, this is no longer a martingale;
- Use the first model with a copula instead of a gaussian distribution.

Who needs hedge funds?
A copula-based approach to hedge fund return replication
H.M. Kat and H.P. Palaro (2005)

Hedge fund replication usually tries to mimic the distribution of the returns of a hedge fund, but hedge funds are often used as an overlay to an already existing portfolio: the authors suggest to also replicate the dependency between the hedge fund and the existing portfolio, as follows.

- Infer the joint distribution of the hedge fund and the existing portfolio using copulas (gaussian, Student, Gumbel, Cook-Johnson (aka Clayton), Frank, symmetrised Joe-Clayton (SJC)) and marginal distributions (gaussian, Student, Johnson S_U);
- Among all the payoff functions, find the cheapest that produces this distribution (this is Dybvig's *payoff distribution pricing model* (PDPM) generalized to a 2-dimensional payoff distribution); it suffices to consider path-independent payoff function (any payoff distribution generated by a path-dependent payoff function can also be generated by a path-independent payoff function); if the Sharpe ratio of the underlying asset is high enough and the correlation with the investor's portfolio low enough, the payoff function will be non-decreasing;
- Price and find a replicating strategy for this payoff function.

Estimation of value at risk using Johnson's S_U -normal distribution
P. Choi (2001)

Value at risk (VaR) estimation can be univariate (you only have the returns of a portfolio) or multivariate (you know the returns of the constituents of your portfolio, *i.e.*, you want to estimate a quantile of $\sum_i w_{it} r_{it}$, where r and perhaps also w is a random variable); it can be conditional (e.g., conditional heteroskedasticity, but you could also have models with conditional third or fourth moments; the article also mentions exponential smoothing (RiskMetrics), which is a special case of IGARCH) or unconditional (extreme value theory (EVT)).

The article studies VaR estimation in a GARCH model with S_U -normal innovations.

A random variable Y is S_U -normal if it is of the form

$$Y = \sinh(\lambda + \theta X)$$

where $\theta > 0$ and X is standard gaussian. Those distributions can be skewed and have fat tails.

Reinforcement learning: a survey
L.P. Kaelbling et al.
Journal of artificial intelligence research (1996)

In *reinforcement learning*, the agent has to make choices and receives some feedback on those choices. Contrary to supervised learning, he is not told which action would have been the best; furthermore, the consequences of an action need not be immediate.

The agent typically tries to maximize one of the following quantities (they do not lead to the same optimal policies):

- N -step optimal control

$$E \left[\sum_{t=n+1}^N r_t \right], \quad n \in \llbracket 0, N \rrbracket$$

- N -step receding horizon control

$$E \left[\sum_{t=n+1}^{n+N} r_t \right]$$

- Infinite-horizon discounted model

$$E \left[\sum_{t=1}^{\infty} \gamma^t r_t \right]$$

- average reward model (this one does not penalize for long learning times)

$$\lim_{N \rightarrow \infty} E \left[\frac{1}{N} \sum_{t=1}^N r_t \right]$$

Asymptotic convergence results are useless: a fast convergence to a near-optimal solution is better than a sluggish convergence to the optimal.

Algorithms have to make a trade-off between exploration (of the possible choices and their consequences) and exploitation.

Dynamic programming can be used to maximize those performance measures.

With a 1-step, immediate-reward model, one can use greedy strategies, *Boltzman exploration* or *interval-based techniques* (for each action, store the number of trials and the number of successes and compute a confidence interval on the probability of success). These algorithms can be adapted to multi-step, immediate-reward problems.

Markov decision processes (MDP) can be used to model delayed reward. If the decision process were known (we do assume that the states and transition probabilities are known, though), we could compute the value of each state for the discounted model and, from there, the optimal policy.

Value iteration estimates the value of each node and derives a strategy (since a strategy is a finite object, it converges in a finite number of steps); *policy iteration* directly estimates the policy.

But we do not know the model.

The *adaptive heuristic critic* algorithm is similar to policy iteration, but the value function is computed iteratively, from the $TD(0)$ algorithm:

$$V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$$

where r is the reward, s the current state, s' the next state. The $TD(\lambda)$ algorithm also updates states that were recently visited.

The *Q-learning* algorithm (if you do not know which algorithm to implement, use this one) estimates $Q(s, a)$,

the expected discounted reinforcement of taking action a while in state s , with the $TD(0)$ or $TD(\lambda)$ algorithm, with a decreasing α . You must first explore the states (but the algorithm is robust) and, after convergence, you can act greedily.

Other algorithms (*certainty equivalence*, *Dyna*, *prioritized sweeping*) learn the model at the same time as the optimal policy: they make a better use of the data and converge faster.

In case of very large state spaces, you can aggregate some states and solve a coarser problem (*multi-grid methods*, *state aggregation*); you can also replace the mappings (transition probabilities, value functions, policies, etc.) by a simpler representation obtained by *supervised learning*. There are other *generalization* algorithms.

Computer science and game theory

J.Y. Halpern

arXiv:cs/0703148

Complexity plays a role in game theory:

- Agents can have *bounded rationality*, i.e., have limited computational power (e.g., they can be finite automata); this can lead to cooperation in the prisoner's dilemma;
- Many problems about *Nash equilibrium* are NP-hard;
- One cannot design a non-dictatorial voting scheme immune to manipulation by voters, but this manipulation can be made computationally unreasonable; there are similar problems in *combinatorial auctions* (auctions where you can bet on bundles of objects; but pricing 2^N objects is too long);
- Agent-based games (e.g., *Byzantine agreement*) are very similar to distributed computing, networking protocols, fault-tolerance, mediator-less cryptography.

This (concise) article also mentions random graphs, bayesian and Markov networks, reinforcement learning.

What every investor should know about commodities

part 1: univariate return analysis

H.M. Kat and R.C.A. Oomen (2006)

Commodities do not offer a risk premium: the average return is not significantly different from zero; the returns depend (slightly) on the business cycle, on monetary conditions, on the futures curve (upward sloping (contango) or downward sloping (backwardation)).

Commodities are not more volatile than stock, as measured by the 2.5%–97.5% range, the standard deviation, the GARCH(1,1) $\gamma = \alpha + \beta$ parameter (it measures the persistence of volatility); they are more volatile in backwardation; energy is more volatile in recessions and less in expansion phases; volatility shocks tend to persist ($\gamma > 0.9$).

Commodity returns are not skewed, when skewness is measured as normalized third moment, as L -moment

$$\frac{X_{.75} - 2X_{.50} + X_{.25}}{X_{.75} - X_{.25}},$$

as the normalized difference between mean and median

$$\frac{\bar{X} - X_{.50}}{\frac{1}{T} \sum_{i=1}^T |x_i - X_{.50}|}$$

or as the GARCH skew parameter.

All commodity returns have fat tails, as measured by the fourth moment, the L -moment

$$\frac{X_{.875} - X_{.625} + X_{.375} - X_{.125}}{X_{.75} - X_{.25}},$$

a ratio of fractile ranges

$$\frac{X_{.975} - X_{.025}}{X_{.75} - X_{.25}}$$

or the number of degrees of freedom of a GARCH(1,1) model with Student innovations.

Commodity returns are autocorrelated.

***What every investor should know
about commodities
part 2: multivariate return analysis
H.M. Kat and R.C.A. Oomen (2006)***

Commodity returns are correlated within groups but not between groups, except “softs”.

Commodity returns are weakly correlated with stock or bond returns, as measured by correlation or Kendall’s tau, but this depends on the business cycle; there is little persistence in correlation dynamics, as measured by a GARCH(1,1)-DCC model, except metals.

There is tail dependence, as measured by the symmetrized Joe Clayton copula (SJC), which has two parameters for tails and head dependence, within groups but not between groups; there is no difference between head and tail dependence: correlation is sufficient.

There is no tail dependence between commodities and stocks or bonds.

Commodity futures are correlated with unexpected inflation, as measured by inflation minus interest rate or change in inflation.

***On more robust estimation of skewness
and kurtosis: simulation and application to
the S&P500 index
T.H. Kim and H. White (2003)***

One can define quantile-based measures of skewness:

$$\begin{aligned} SK_1(\alpha) &= \frac{X_{1-\alpha} - 2X_{.5} + X_{\alpha}}{X_{1-\alpha} - X_{\alpha}} \\ SK_2 &= \frac{\int_0^{1/2} (X_{1-\alpha} - 2X_{.5} + X_{\alpha}) d\alpha}{\int_0^{1/2} (X_{1-\alpha} - X_{\alpha}) d\alpha} \\ &= \frac{E[X] - X_{.5}}{E[|X - X_{.5}|]} \\ SK_3 &= \frac{E[X] - X_{.5}}{\sigma(X)} \end{aligned}$$

(SK_3 is not compatible with “skewness ordering” – but this notion is not defined) and kurtosis (those measures are usually corrected by subtracting the value for a gaussian distribution):

$$\begin{aligned} KR_1 &= \frac{(X_{7/8} - X_{5/8}) + (X_{3/8} - X_{1/8})}{X_{3/4} - X_{1/4}} \\ KR_2(\alpha, \beta) &= \frac{E[X|X > X_{1-\alpha}] - E[X|X < X_{\alpha}]}{E[X|X > X_{1-\beta}] - E[X|X < X_{\beta}]} \\ KR_2(\alpha, \beta) &= \frac{X_{1-\alpha} - X_{\alpha}}{X_{1-\beta} - X_{\beta}} \end{aligned}$$

(with $\alpha = 0.05$, $\beta = .5$ in KR_2 and $\alpha = 0.025$, $\beta = .25$ in KR_3).

***Significance tests harm progress in forecasting
J.S. Armstrong (2005)***

Significance tests should be avoided:

- readers and authors misinterpret them;
- the choice of the null hypothesis is problematic;
- there is a publication bias against non-rejection of the null;
- it distracts the authors from other issues: they often fail to assess importance (with *effect sizes* or confidence (with *confidence intervals*)).

Furthermore, if you have enough data, everything is statistically significant.

***Forecasting with many predictors
J.H. Stock and M.W. Watson (2005)
Handbook of economic forecasting***

The authors review several classes of forecasting algorithms: forecast combination (without any discussion of bagging or boosting), bayesian model averaging (BMA, rarely used, but better than an equal-weighted combination of predictors), *empirical Bayes methods* (the prior is not a prior, it comes from the data) and dynamic factor models (no mention of biased estimators or regularization paths).

In a *dynamic factor model* (DFM), unobserved AR factors \mathbf{f} and their lags explain the observed variables X_{it} :

$$\begin{aligned} \mathbf{f}_t + \Gamma_1 \mathbf{f}_{t-1} + \dots + \Gamma_k \mathbf{f}_{t-k} &= \boldsymbol{\eta}_t \\ X_{it} &= \lambda_{i0} \mathbf{f}_t + \dots + \lambda_{i\ell} \mathbf{f}_{t-\ell} + u_t \end{aligned}$$

while a (static) factor model would be

$$X_{it} = \lambda_i f_t + u_{it}.$$

Dynamic factor models can be reformulated as static factor models and estimated by maximum likelihood; there are also principal-component-analysis-based approximations.

The article interprets this model in terms of the *spectral density matrix* without bothering to define it.

Dynamic principal component analysis (PCA) is similar; the corresponding latent variables can be obtained by performing a PCA dimension reduction on the spectral density.

Principal components at work: the empirical analysis of monetary policy with large datasets
C.A. Favero et al. (2002)

Comparison of two estimators of *dynamic factor models*, based on static (time-domain) and dynamic (frequency-domain) PCA (principal component analysis): the performance is similar, but dynamic PCA is more parsimonious.

Moment problems via semi-definite programming: applications in probability and finance
P. Popescu and D. Bertsimas (2000)

The Markov, Chebychev, Chernoff inequalities provide bounds on some probabilities $P[X \in \Omega]$ given some moments or joint moments of arbitrary random variables. More generally, one can try to solve the following optimization problem:

$$\begin{aligned} &\text{Maximize } E[\phi(X)] \\ &\text{such that } \forall i \quad E[f_i(X)] = q_i \end{aligned}$$

for all random variable $X : \Omega \rightarrow \mathbf{R}^m$.

For instance, in finance, one can look for bounds on the price of an option given the moments of the underlying and/or other option prices (we only have to assume that the option price is the expected discounted payoff under the risk-neutral measure; we do not know anything about that measure except that it is a probability measure).

The *feasibility* of that optimization problem can be expressed in terms of positive semi-definiteness – e.g., in a multi-dimensional context, the second centered moment (the variance matrix) should be positive semi-definite.

In the *dual* problem (under reasonable conditions, there is a strong duality theorem), the expectation disappears but we have an infinite number of constraints:

$$\begin{aligned} &\text{Minimize } y'q \\ &\text{such that } \forall x \in \Omega \quad y'f(X) \geq \phi(x). \end{aligned}$$

Fortunately, those constraints can often be expressed as semi-definite constraints and the problem can be solved.

In higher dimensions, beyond the second moment, the problem of finding optimal bounds is NP-hard.

This is a review article: there are more detailed publications by the same authors.

Second order cone programming approaches for handling missing and uncertain data
P.K. Shivaswamy et al.
Journal of machine learning research (2006)

In the support vector machine (SVM) binary classification optimization problem

$$\begin{aligned} &\text{Minimize } \frac{1}{2} \|z\|^2 + C \sum_i \xi_i \\ &\text{such that } \forall i \quad y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i \\ &\quad \forall i \quad \xi_i \geq 0 \end{aligned}$$

uncertainty can be introduced by transforming the constraints into

$$P[y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i] \geq 1 - \kappa_i.$$

The *Chebychev inequality* gives a worst-case bound on this probability, provided you know the first two moments of x .

This can be generalized to classification into more than two groups and to regression.

Applications of second order cone programming
M.S. Lobo et al. (1998)

Many optimization problems can be recast as second order cone programs (SOCP): quadratically-constrained quadratic programs (QCQP), optimization problems in which the objective is a sum or max of norms; FIR (finite impulse response) filter design, robust linear programming, robust least squares, robust portfolio optimization.

Interior-point methods adapted to SOCP are faster than those adapted to the more general semidefinite programs (SDP).

Second-order cone programming
F. Alizadeh and D. Goldfarb (2002)

More detailed article on the same subject.

Introducing Mr. Risk
A. Meucci

Very simple (but readable and rigorous) article about portfolio construction: variance matrix (*i.e.*, risk model) construction, optimization, warning on the confusion between linear and log-returns, myopic allocation (multi-period asset allocation in the special case where the weights are constant), portfolio insurance via delta hedging, decomposition of homogeneous risk measures (e.g., the value at risk) using *Euler's formula*.

The last chapter presents *sensitivity analysis*: after choosing a few risk factors, you can express the future price of an asset as a function of those risk factors; this can be approximated by a Taylor expansion,

$$\Delta \text{price} = \frac{\partial \text{price}}{\partial \text{risk}} \Delta \text{risk} + \frac{1}{2} \Delta \text{risk}' \frac{\partial^2 \text{price}}{\partial \text{risk}^2} \Delta \text{risk}.$$

For instance, in fixed income, you can use time and yield as risk factors; the corresponding derivatives are called *carry* and *duration*

$$C = \frac{\partial \pi}{\partial t} \quad D = -\frac{1}{\pi} \frac{\partial \pi}{\partial y};$$

this leads to the *duration approximation*

$$\Delta \pi \approx C \Delta t - \pi D \Delta y.$$

Similarly, the *delta approximation* is a first-order approximation of the price change of a (vanilla) option using stock indices.

Second-order approximations can be used to compute the value at risk (VaR) of a portfolio.

The article ends with a detailed example: the risk of a portfolio of mortgages.

Caml trading: experiences in functional programming on Wall Street
T. Minsky
The monad reader (2007)

Benefits of functional languages in finance.

Financial econometrics
A.W. Lo (2007)

This soon-to-be-published book (this was only the preface) is a collection of old articles. The table of contents is interesting, though:

- Models of asset returns: time-varying moments, fat tails, long memory, regime shifts, cointegration;
- Asset pricing models: CAPM and its tests, multifactor models (APT), performance measurement, tests of market timing or security selection ability;
- Dynamic asset pricing models: variance bounds controversy (?), equity premium puzzle, consumption-based CAPM, stochastic discounted factor model, incomplete markets, term structure;
- Market microstructure: price discovery (Walrasian tatonnement, etc.), price discreteness, irregular trading intervals, bid-ask spread, binomial “trees”, state-space density estimation, price jumps, serial correlation, functional central limit theory (?)’
- Non-standard finance: survivorship bias, data snooping, Bayesian methods, event study, generalized method of moments, robust statistics, nonlinear dynamical systems, neural networks, technical analysis, random matrix theory.

A jigsaw puzzle of basic risk-adjusted performance measures
H. Scholz and M. Wilkens
Journal of performance measurement (2005)

Performance measures based on total risk

- Sharpe ratio;
- RAP (risk-adjusted performance or *Modigliani’s* M^2): expected returns of the portfolio, leveraged so that its volatility equals that of the market portfolio;
- DR^{RAP} (differential returns based on RAP): difference in expected returns between the market portfolio and the portfolio, leveraged to have the same volatility as the market portfolio;
- TRA (total risk alpha): difference in expected returns between the portfolio and the market portfolio, leveraged to have the same volatility as the portfolio studied;

can be modified by replacing the total risk σ by the market risk β :

- Treynor ratio;
- MRAP (market-risk-adjusted performance);
- DR^{MRAP} (differential returns based on MRAP);
- Jensen’s alpha.

The authors explain those definitions with pictures.

Symmetric versus asymmetric conditional covariance forecasts: does it pay to switch?
S. Thorp and G. Milunovich
Journal of financial research (2007)

Multidimensional, asymmetric GARCH models in portfolio optimization can lower realized portfolio risk.

Model comparison of coordinate-free multivariate skewed distributions with an application to stochastic frontiers
J.T.A.S. Ferreira and M.F.J. Steel
Journal of econometrics (2007)

Most multivariate families of distributions used to model skewness are not coordinate-free: the coordinate axes play a special role, skewness is only allowed in some (coordinate-dependent) directions – those families are not closed under orthogonal transformations.

Coordinate-free multivariate skew distributions can be obtained as follows: choose n univariate skewed distributions, apply an invertible n -dimensional affine transformation.

Univariate skewed distributions can be obtained as follows, from a symmetric distribution with pdf (probability distribution function) f and cdf (cumulative distribution function) F :

- Hidden truncation: $s(x) \propto f(x)F(\alpha x)$;
- Inverse scale factor (scale the distribution by a different factor depending on the sign): $s(x) \propto f(\gamma^{-\text{sign } x} x)$;
- More generally, one can consider $s(x) = f(x)p(F(x))$

for various choices for p , e.g.,

$$p(u) = \frac{u^{\tau-1}(1-u)^{1/\tau-1}}{B(\tau, 1/\tau)}.$$

***Enjoy the joy of copulas*
J. Yan (2006)**

Presentation of the `copula` R package to sample from, plot or fit gaussian, Student, Clayton, Frank or Gumbel copulas in any dimension.

***Analyst recommendations, mutual fund herding and overreaction in stock prices*
N.C. Brown et al. (2007)**

Mutual funds herd after consensus analyst changes – exploiting their overreaction can earn you 6% per year.

***A robust version of the KPSS test based on indicators*
R.M. de Jong
Journal of econometrics (2006)**

In the KPSS test, instead of demeaning the data $x_i \leftarrow x_i - \bar{x}$, just use $x_i \leftarrow \text{sign}(x_i - \text{median } x)$: this does not change the asymptotic distribution, relaxes assumptions about the moments and increases the power of the test in presence of fat tails (but reduces it otherwise).

***Multivariate archimedian copulas, d -monotone functions and ℓ_1 -norm symmetric distributions*
A.J. McNeil and J. Nešlehová**

Archimedian copulas are exactly the ℓ_1 -symmetric copulas that avoid the origin.

The article also characterizes functions that generate d -dimensional archimedian copulas.

***On the interaction between ultra-high frequency measures of volatility*
G.M. Gallo and M. Velucchi (2007)**

The realized variance (*i.e.*, variance estimated from tick data) $\sum x_t^2$ and the *bipower variance* $\sum |x_t x_{t-1}|$, robust to jumps, are almost undistinguishable in the context of *multiplicative error models* (MEM), *i.e.*, GARCH-like models used to model positive quantities (volume, duration, range, etc.).

***Estimating copula functions from limited data*
D. Allwright et al. (2006)**

The problem of estimating the value at risk (VaR) of a random variable without any distributional assumption (or when the distribution has too many parameters, say, a 2-parameter Pareto distribution or a mixture of gaussians or exponentials) remains the same in higher dimension: it is impossible when $p^n > \alpha$, where p is the quantile to estimate, $1 - \alpha$ is the confidence level and n the number of observations (say, $p = 0.99$, $\alpha = 0.05$).

***Zero-intelligence realized variance estimation*
J. Gatheral and R.C.A. Oomen (2007)**

Realized variance estimators (*i.e.*, variance estimators using high frequency data) should avoid market microstructure artefacts such as the *bid-ask bounce* – one can do better than aggregate the data, *i.e.*, throw away 99% of it. The article compares twenty such estimators on simulated data (using the *zero-intelligence limit order book model*: orders arrive at random, following a Poisson distribution, and are randomly cancelled). Their conclusions are:

- Do not forget to correct for any bias in your estimator;
- Prefer mid-quotes to trade prices, they are much less noisy;
- Use subsampling or kernel estimators (a *kernel variance estimator* is a linear combination of $\text{Cov}(x_t, x_t)$, $\text{Cov}(x_t, x_{t-1})$, $\text{Cov}(x_t, x_{t-2})$, etc., with well-chosen decreasing coefficients).

***Issues in creating a sentiment measurement based on internet stock message boards*
Y.C. Zhang and P.E. Swanson**

Internet message boards (thelion.com, Yahoo finance, the motley fool, raging bull, etc.) can be used to build a sentiment index; reply messages should be given a lower weight (0.125); the *maximum entropy text classifier* (not clearly presented) performs better than the naive Bayes one.

***Some methods for training mixtures of experts*
P. Moerland (1997)**

Ensemble learning algorithms typically combine several learning algorithms (or *experts*) using fixed weights (the weights were determined from the training sample but they do not change and depend on each new observation).

On the contrary, you could have several experts controlled by a *gating expert* assigning weights to each of them, depending on the observation to classify.

***Voting with a parametrized veto strategy: solving the KDD cup 2006 problem by means of a classifier committee*
D. Tikk et al.
SIGKDD Explorations (2006)**

Combine several (neural-network-based) supervised learning techniques with a committee vote, using weights from the average performance of a 10-fold cross-validation test and the confidence value of each algorithm, with a veto to control the maximum allowed error rate.

***Maximizing classifier utility when training data is costly*
G.M. Weiss and Y. Tian
SIGKDD Explorations (2006)**

Formalization of what emergency medicine already does – for instance, this is one of the motivations of *decision trees*, which use *few* variables.

***Portfolio management using value at risk
A comparison between genetic algorithms
and particle swarm optimization*
V.A.D. Filho (2006)**

Non-linear, multi-modal optimization problems, such as value-at-risk (VaR) portfolio optimization (the article is limited to 50 assets) can be solved by *particle swarm optimization* (PSO) or *genetic algorithms* (GA).

In PSO, particles (portfolio weights vectors) fly around and their velocity is updated from their previous velocity (with a linearly decreasing inertia), the best position encountered so far by the particle and the whole swarm:

$$v_{i,t+1} = wv_{i,t} + c_1r_1(\text{best}_i - x_i) + c_2r_2(\text{best}_{\text{swarm}} - x_i) \\ r_1, r_2 \sim U(0, 1).$$

Constraints can be taken care of in different ways:

- *bumping*: when a particle hits a boundary, its velocity is set to zero (but the particles may become trapped near local extrema near the boundary);
- *random positioning*: when a particle hits a boundary, it is put somewhere else, randomly;
- *amnesia*: when a particle hits a boundary, it continues as if there were no boundary but stops recording the best position so far;
- *penalty* function.

In the GA, individuals are selected for reproduction by a *roulette wheel* (with a probability proportional to their fitness) or by a *tournament* (pairs of individuals are selected at random and the best one is chosen); the crossover can be *basic*

$$(x, y) \mapsto (x_{1..k} \sqcup y_{k+1..n}, y_{1..k} \sqcup x_{k+1..n})$$

or *arithmetic*

$$(x, y) \mapsto (px + (1 - p)y, (1 - p)x + py);$$

the mutation operator is

$$(x_k, x_\ell) \leftarrow (x_k + r, x_\ell - r) \\ k, \ell \sim U(\llbracket 1, n \rrbracket) \\ r \sim U(0, 1).$$

PSO is faster but GA are more robust.

***Stochastic analysis of an agent-based model*
A. Veglio and M. Marsili
arXiv:0705.4025**

An agent-based herding model.

***Extending Black–Litterman analysis beyond
the mean-variance framework*
L. Martellini and V. Ziemann
Journal of portfolio management (2007)**

The authors extend the Black–Litterman framework to take into account skewness and kurtosis: the simply replace the benchmark portfolio (used to compute the implied alpha) by the minimum VaR (value at risk) portfolio, where the VaR is computed from a Cornish–Fisher expansion and the CAPM (capital asset pricing model) is replaced by a 4th moment CAPM. The classical Black–Litterman formula is then applied.

This is an approximation of Meucci’s generalization: he considered the whole returns distribution, not just its first moments.

***On dynamic measures of risk*
J. Vcitanic and I. Karatzas (1998)**

An agent has some initial capital x , some liability C to be met at time T and tries to find a strategy π so that his wealth at time T exceeds the liability:

$$X^{x,\pi}(T) \geq C.$$

He can do so provided the initial capital x is large enough,

$$x \geq E_{\text{risk-neutral}} \left[\frac{C}{S_0(T)} \right]$$

(where S_0 is the price of the risk-free asset). If not, its risk can be measured as

$$\inf_{\pi} E_{\text{real-world}} \left[\frac{C - X^{x,\pi}(T)}{S_0(T)} \right]^+$$

or even, to account for model uncertainty,

$$\sup_{P \in \mathcal{P}} \inf_{\pi} E_P \left[\frac{C - X^{x,\pi}(T)}{S_0(T)} \right]^+.$$

***The Epps effect revisited*
B. Tóth and J. Kertész
arXiv:0704.1099**

Stock correlations decrease as the sampling frequency increases: this is the *Epps effect*. The classical explanations:

- the maximum cross-correlation is the same but is not attained at a zero time lag;
- asynchronicity of ticks

are not sufficient. The article provides a physicist’s explanation for the remaining effect.

Also check arXiv:0704.3798, by the same authors.

***Price discovery in the US stock and stock
options markets: a portfolio approach*
R. Holowczak et al. (2006)**

Many articles try to guess the direction of the information flow between the stock and option markets, *i.e.*, in

which market price discovery occurs (with inconclusive results).

To remove the non-linear effects of options, the authors consider synthetic forwards, *i.e.*, portfolios made of a put-call pair, with a linear payoff.

Price discovery seems to occur in stock markets, except in case of unusual option trading activity.

Structural breaks in financial time series **E. Andreou and E. Ghysels (2006)**

Not accounting for structural changes can lead to over-complicated models.

Here are some partial-sum-based (CUSUM) statistics to test for a structural break:

- Fluctuation = $\frac{\bar{X}_{[m+1,n]} - \bar{X}_{[1,m]}}{\sigma/(n-m)}$;
- PS = $\sum_{i=m+1}^{m+k} (X_i - \bar{X}_{[1,m]})$;
- PG = $\sum_{i=1}^n X_i - \text{Min}_{1 \leq i \leq n} \sum_{i=1}^n X_i$.

The asymptotic framework developped by Kuan, Hornik and Leisch to detect structural changes in linear regressions can be adapted to non-linear processes (GARCH, etc.) and can help determine various thresholds beyond which those partial sum statistics should be considered suspicious.

To spot multiple breaks, one can use *penalized likelihood* methods or, for stochastic volatility (GARCH, etc.) models, *minimum description length* (MDL)-based tests (no details given).

The bulk of the article is devoted to checking the conditions under which those tests are valid.

Notes sur les mesures de performance **D. Herlemont** **YATS (2004)**

The *Sharpe ratio* is not compatible with *strong dominance* and can be manipulated (just write out-of-the-money puts); non-gaussian arbitrage oportunities can have an arbitrarily small Sharpe ratio.

A *Bonnet–Nagot risk measure* π satisfies:

- $X \perp\!\!\!\perp Y \implies \text{Max}_{\alpha, \beta} \pi(\alpha X + \beta Y) = \pi(X) + \pi(Y)$
- The maximum is reached for $\alpha = \lambda(X)$ and $\beta = \lambda(Y)$ where $\lambda(Z)$ only depends on Z .

Up to an affine transformation, a Bonnet–Nagot measure of risk coincides with the Sharpe ratio for gaussian distributions.

They can be characterized in terms of the log-Laplace transform $H_X(\lambda) = -\ln Ee^{-\lambda X}$ as

$$\pi(X) = \sup_{\lambda} \mathcal{J}(H_{\lambda X})$$

where $\mathcal{J} : \mathcal{C}^\infty(\mathbf{R}, \mathbf{R}) \longrightarrow \mathbf{R}$ is linear and continuous.

Examples include:

- The Sharpe ratio:

$$\mathcal{J}_{\text{SR}}(H) = H'(0) + \frac{1}{2}H''(0)$$

- The *Hodges* (or *Stutzer*) measure:

$$\mathcal{J}_{\text{Hodges}}(H) = H(1)$$

- The Bonnet–Nagot measure:

$$\mathcal{J}_{\text{BN}}(H) = H(1) + aH(\alpha).$$

(The link with Acerbi's *coherent* measures of risk is mentionned but not explained.)

The Stutzer measure is actually the rate of convergence to zero of the ruin probability (studied in *large deviation theory*); the Stutzer criterion is a (riskier) alternative to the Kelly criterion, which maximizes the rate of growth instead.

Information uncertainty and stock returns **X.F. Zhang (2004)**

Greater information uncertainty produces higher returns following good news and lower returns following bad news; “news” are measured by forecast revisions or short-term momentum; uncertainty is measures by firm size, fir age, analyst coverage, analyst forecast dispersion, returns volatility, cash flow volatility. In other words: the more volatile or uncertain the stock, the better momentum works.

The underlying dynamics of credit correlations **A. Berd et al.** **Journal of credit risk (2007)**

The default of a basket of companies is usually modelled as follows:

- Consider boolean, time-dependent *loss variables* $Y_{it} = 1_{R_{it} \geq d_{it}}$, where R_{it} are *latent* variables and d_{it} thresholds (e.g., $R_{it} = R_t$ is the time of default of company i and $d_{it} = t$);
- For fixed t , study the *copula* of the latent variables (use a 1-factor gaussian copula);
- Consider the *loss distribution*, *i.e.*, the distribution of $\sum_i \ell_i Y_i$.

The *correlation surface* $\rho(K, p)$ is the correlation parameter of the gaussian copula that produces the same expected loss for the tranche $[0, K]$ and default probability p .

Pricing forward contracts in power markets by the certainty equivamence principle: explaining the sign of the market risk premium **F.E. Benth et al. (2006)**

Explanation of the risk premium for commodities, from a 2-agent model (consumer and producer), each with a different utility, and an underlying with a stochastic price.

Asset price dynamics with heterogeneous groups

G. Caginalp and H. Merdan
Physica D (2007)

Asset price dynamics can be modeled by a system of differential equations, similar to the predator-prey systems in biology.

Generalized canonical regression
A. Estrella (2007)

Linear regression assumes that there is a single dependent variable

$$y = \sum_i \beta_i x_i + \varepsilon;$$

with several dependent variables, one can use *multiple regression*

$$\forall j \quad y_j = \sum_i \beta_{ij} x_i + \varepsilon_j$$

(often with some added structure to account for dependencies between the y_j) or *canonical regression*:

$$\sum_j \alpha_j y_j = \sum_i \beta_i x_i + \varepsilon$$

(the left-hand side is “the most predictable linear combination”).

The traditional assumptions of canonical regression are too restrictive: this article relaxes them and derives the asymptotic distribution of the coefficients, in order to design statistical tests.

Forwards and European options on CDO tranches

J. Hull and Al. White
Journal of credit risk (2007)

Options on CDOs, once the parameters (default probability, expected recovery rate) have been estimated, are not that different from vanilla European options.

The article also clearly recalls what CDS and CDO are:

- In a *credit default swap* (CDS), the buyer makes periodic payments (quarterly in arrears) until default or maturity; in case of default (credit rating change, delayed payment, bankruptcy, etc.), the sellers pays the notional value times one minus the recovery rate (and the buyer makes a final accrual payment);
- A *collateralized debt obligation* (CDO) is a claim on a portfolio of CDS; the portfolio is cut into *tranches* (the order statistics of the CDS);
- iTraxx are CDO tranches on 125 European firms;
- CDX are CDO tranches on 125 US firms.

Modelling asymmetric exchange rate dependence
A.J. Patten (2005)

Conditional copulas, i.e., the colula of $XY|Z$, can be used to assess how the relation between two variables depends on a third.

(Instead of considering a copula depending on a an exogenous variable, one could consider a higher-dimensional copula – but the exogenous variable likely plays a very different role.)

The performance of model-based option trading strategies
B. Eraker

Option pricing models should not be assessed by comparing predicted and actual prices but by looking at the profitability of the corresponding strategy (in efficient markets, these would be equivalent).

The weights in the resulting strategy depend on the price difference and its sensitivity to estimation errors – this is an optimization problem.

Improved option pricing using artificial neural networks and bootstrap methods
P.R. Lajbcygier anf J.T. Connor
International journal of neural systems (1997)

If you are unsatisfied with your option pricing model, just try to predict its residuals with a neural network, a bootstrap or a bagging algorithm.

Non-parametric determination of real-time lag structure between two time series: the “optimal thermal causal path” method

D. Sornette and A.-X. Zhou
arXiv:cond-mat/0408166

The time-warping algorithm can be generalized in a fuzzy (Boltzmann) way to make it robust to noise.

The authors apply these results to the study of causal relations between macroeconomic time series.

On developing a financial prediction system: pitfalls and possibilities
S. Zemke (2002)

A bullet-point list of *machine learning* ideas to be applied to finance.

Model uncertainty and forecasting, a practitioner point of view
B. Bellone and E. Michaux (2006)

The article examined several ways of tackling model uncertainty:

- Linear regression bayesian model averaging (BMA), good in the short term;
- Dynamic factor models (DFA, i.e., regression on the factors of a factor analysis and their lags), good in the long term;
- Median BMA (model obtained from the variables with an inclusion probability beyond 1/2);
- General-to-specific (GETS) algorithm (a stepwise variable selection, with multiple steps and tests).

The authors use a Scilab econometric toolbox, “Grocer”, developed at the ENSAE.

***Momentum, reversal
and the trading behaviors of institutions***
R.C. Gutierrez and C.A. Prinsky
Journal of financial markets (2007)

Cumulated residual returns (from a 5-year CAPM model) can help devise a long-term profitable strategy, while the other component of the momentum (“relative momentum”) reverts after one year.

The article builds a long-short portfolio on the signal

$$\alpha = \frac{\text{cumulated residual returns}}{\sqrt{\text{cumulated variance of the residual returns}}}.$$

This effect is linked to institutional ownership.

***Macroeconomic factors
and Japan’s industry risk***
P. Nguyen
**Journal of multinational financial management
(2007)**

The *risk* of an industry can be defined as the average cash flow, weighted by firm size, divided by the (equal-weighted) volatility of those cash flows.

***Analyzing digits for portfolio formation
and index tracking***
P.N. Posch and W.A. Kreiner

The Newcomb–Benford law (or *first-digit law*),

$$P[X_1 = d] = \log_{10} \left(1 + \frac{1}{d} \right), \quad d \in \llbracket 1, 9 \rrbracket$$

is already used to detect fraud in tax sheets and company reports.

The authors put this distribution in a 1-parameter family

$$P_d(\lambda) \propto (d+1)^{1-\lambda} - d^{1-\lambda}$$

and compute this parameter, the *digital-fit factor* (DFF) for the price of the members of an index: it moves in the opposite direction as the index (the incorrectly use correlation to assess this – the time series are integrated).

They then try to use it to design an investment strategy, but their strategy is merely a leverage-increasing strategy, and they even forget to use their DFF...

***A statistical derivation
of the significant-digit law***
T.P. Hill
Statistical science (1996)

(Theoretical article on Benford’s Law.)

***Synchronizing multivariate
financial time series***
F. Audrino and P. Bühlmann (2001)

When combining data from markets in different time zones, care should be taken: two time series X , Y from

(say) New York and London can be seen as sampled at two different times, London close and New York close, with every other value missing. This is a missing value problem: models can be fitted using the EM algorithm.

The article applies this idea to the CCC-GARCH(1,1) model.

Alternatively, one can use one data point per day at a time when both markets are opened.

With weekly data, the problem disappears.

***Correlations and volatilities
of asynchronous data***
P. Burns et al. (1998)

Earlier article on the same subject.

***Why risk and return are uncorrelated:
a relative status approach***
E. Falkenstein (2006)

Expected utility models where utility is a function of the wealth relative to other market participants predict there is no risk premium.

Long-term dependence in stock returns
J.T. Barkoulas and C.F. Baum

Long memory (or “fractal dynamics”) can be tested using:

- Hurst’s rescaled-range (R/S) analysis;
- Lo’s modified R/S analysis;
- Geweke’s spectral regression: in an ARFIMA model,

$$\Phi(L)(1-L)^d(Y_t - \delta) = \Theta(L)\varepsilon_t$$

$$(1-L)^d = \sum_{k \geq 0} \frac{\Gamma(k-d)}{\Gamma(-d)\Gamma(k+1)} L^k$$

the fractional integration parameter d can be estimated as the opposite of the slope in the regression

$$\ln I(\xi) \sim \ln \sin^2 \frac{\xi}{2}$$

where $I(\xi)$ is the periodogram of y at frequency ξ .

Some stocks exhibit long memory, but the effects disappear when they are aggregated into stock indices.

***Residual-based tests for fractional
cointegration: a Monte Carlo study***
I. Dittmann (1998)

Comparison of many *fractional* cointegration tests (Geweke, modified rescaled range, ADF, etc.): prefer the Phillips–Perron T -test, it is usually more powerful.

Symbolic representation of long time-series
G. Hébrail and B. Huhueney

To plot a very large and somewhat periodic time series, perform a fast Fourier transform (FFT) to find the

largest period, cut the time series accordingly, cluster the chunks with a self-organizing map (SOM) and use a different icon for each cell.

***Dynamic portfolio selection
by augmenting the asset space***
M.W. Brandt and P. Santa-Clara
Journal of finance (2006)

You can replace the (difficult) choice of a dynamic strategy by a static choice of managed portfolios; the authors use a constant linear combination of “state variables” and maximize

$$E \left[(\theta z)' r - \frac{\gamma}{2} (\theta z)' r r' (\theta z) \right]$$

where z are the state variables, θ their weights, θz the weight of the stocks, r the forward returns, γ the risk aversion.

***On default correlation:
a copula function approach***
D.X. Li
RiskMetrics (2000)

(Non-gaussian) copulas should be used to model the joint distribution of time-until-default (some people use a binary variable indicating a default in the next year, which is very imprecise) instead of the correlation (as CreditMetrics does).

This can be used to price credit default swaps or first-to-default swaps.

Analyzing active investment strategies
M. Ammann et al.
Journal of portfolio management (2006)

Analysis of variance (anova) can help find out which strategy (“style”) an asset manager is following.

Managing guarantees
M.A.H. Dempster
Journal of portfolio management (2006)

The methods of asset-liability management (ALM), *i.e.*, dynamic stochastic programming, can be used to implement guaranteed strategies, thereby competing with portfolio insurance.

The technical details might be “a challenge even for sophisticated users”.

Pairs trading: the new generation
S. Hartman et al.
ABN Amro (2006)

A pairs trading strategy can be improved by selecting pairs with attractive fundamentals (or even playing them against pairs with poor fundamentals).

***A liquidity-augmented
capital asset pricing model***
W. Liu

Journal of financial economics (2006)

The CAPM (capital asset pricing model) overlooked the *liquidity premium*.

To account for all the dimensions of liquidity (quantity, speed, cost, price impact – speed is often forgotten), the author develops yet another measure:

$$LM_x = \left(n_x + \frac{\lambda_x}{\text{turnover}_x} \right) \frac{21x}{N_x}$$

where $x = 1, 2, 12$ months, n_x is the number of zero daily trading volume in the past x months, N_x is the number of trading days in the past x months, turnover_x is the sum of the daily turnover (number of shares exchanged divided by the number of outstanding shares) over the past x months, λ_x is chosen so that

$$0 < \frac{\lambda_x}{\text{turnover}_x} < 1$$

for all stocks.

***Equity portfolios generated by functions
of ranked market weights***
R. Fernholz (Intech, 2000)

Functionally-generated portfolios have weights of the form

$$w_i = f(\text{MCap}_i).$$

They can be generalized to

$$w_i = f(\text{rank}(\text{MCap})_i).$$

***Parameters for estimation of entropy to study
price manipulation in stock market***
Y.V. Reddy and A. Sebastin

Confusing article presenting various entropy estimators (approximate entropy, sample entropy) and applying them to price, time, quantity data.

***Portfolio choice with jumps:
a closed form solution***
Y. Aït-Sahalia et al. (2006)

By imposing a factor structure on the asset returns, one can derive a closed form solution to the portfolio choice problem for Levy returns.

(The article is technical, I have not read the details.)

***Forecasting exchange rates:
a robust regression approach***
A. Preminger and R. Franck (2005)

Robust regression can also be used to fit autoregressive models.

***The PIN anomaly
around M&A announcements***
N. Aktas et al.
Journal of financial markets (2006)

The *probability of informed trading* (PIN) can be estimated from the following model (add a few independence assumptions), via maximum likelihood:

- α : probability of information availability on a given day;
- δ : probability of a positive price change after an event;
- μ : rate of arrival of uninformed buyers and sellers;
- ε : rate of arrival of informed traders;
- $$\text{PIN} = \frac{\alpha\mu}{\alpha\mu + 2\varepsilon}$$

***Fund manager use of public information:
new evidence on managerial skills***
M. Lacperczyk and A. Seru
Journal of finance (2007)

The *reliance on public information* (RPI), defined as the R^2 of the regression of holdings changes against analyst recommendation changes, is a measure of a portfolio manager's lack of skill.

***On the use of data envelopment analysis in
assessing local and global performances of
hedge funds***
H. Nguyen-Thi-Thanh (2006)

Yet another article on the applications of data envelopment analysis (DEA) to hedge fund comparison – previous ones forgot the fees.

***On the relationship between changes in stock
prices and bond yield in the G7 countries:
wavelet analysis***
S. Kim and F. In
**Journal of international financial markets,
institutions and money (2007)**

Wavelet correction analysis (a time series of length N yields wavelet coefficients that can be arranged in a pyramid, $N/2$ coefficient on the first row, $N/4$ on the second, etc.; one can then compute the variance (and covariance, if you have several time series) of each of these rows) allows the authors to examine short- and long-term relations between stock and bond returns: except in Japan, they are negatively related.

***Mean-semivariance behavior: downside risk
and capital asset pricing***
J. Estrada
**International review of economics and finance
(2007)**

The CAPM (capital asset pricing model) can be augmented by a downside risk:

$$\text{return} = \alpha + \beta \cdot (\text{market returns}) + \beta_- \cdot (\text{market returns})_-$$

The authors prefer to *replace* the beta β by the downside beta β_-

***The fundamentals
of commodity futures returns***
G.B. Gorton et al (2007)

Inventories have a non-linear effect on the risk premium.

***Beyond the gaussian copula:
stochastic and local correlation***
X. Burtschell et al.
Journal of credit risk (2007)

People still use the *1-factor gaussian copula*

$$\begin{aligned} \text{Cor}(X_i, X) &= \rho \\ i \neq j &\implies X_i|X \perp\!\!\!\perp X_j|X \end{aligned}$$

to price CDO tranches. The authors suggest a *stochastic correlation model* with a gaussian copula of the form

$$\begin{aligned} \text{Cor}(X_i, X) &= \rho_i \\ i \neq j &\implies X_i|X \perp\!\!\!\perp X_j|X \\ \rho_i &\in \{0, \rho, 1\} \end{aligned}$$

where the ρ_i are discrete random variables.

***Extreme returns from extreme value stocks:
enhancing the value premium***
K. Anderson and C. Brooks
Journal of investing (2007)

The relation between the P/E and future returns is not linear: considering quintiles is not sufficient. The authors use the *idiosyncratic P/E*, defined as the current price divided by the earnings over the past 8 years, corrected for the influence of market, size and sector.

***Residual income approach
to equity country selection***
S. Desrosiers et al.
Financial analysts journal (2007)

Valuation methods usually applied to companies (P/E, ERP, etc.) can also be applied to countries.

Dynamic nonmyopic portfolio behavior
T.S. Kim and E. Omberg
Review of financial studies (1996)

In the *Kelly principle* (see below, Herlemont (2004)), one can replace growth rate maximization by expected utility maximization.

Portfolio selection with transaction costs
M.H.A. Davis and A.R. Norman
Mathematics of operations research (1990)

The Kelly strategy, in continuous time, with transaction costs.

Investment performance measurement, risk tolerance and optimal portfolio choice
M. Musiela and T. Zariphopoulou (2007)

In this and related articles, the second author advocates that utility or other satisfaction measures should be time-dependent – *forward utility, dynamic performance*, etc.

Revisiting calendar anomalies in Asian stock markets using a stochastic dominance approach
H.H. Lean et al.
Journal of multinational financial management (2006)

First order *stochastic dominance* for the *January effect* has disappeared; weekly and monthly seasonality is still present (there are statistical tests for stochastic dominance, such as the Davidson and Duclos test).

Stochastic dominance is a very strong result, but it captures effects not visible if you only look at the first two moments.

Mean reversion versus random walk in G7 stock prices: evidence from multiple trend break unit root tests
P.K. Narayan and R. Smyth
Journal of international financial markets, institutions and money (2007)

Even after accounting for multiple breaks in the trend, index prices are not stationary.

Inference in inequality from household survey data
D. Bhattacharya (2006)
Journal of econometrics (2007)

Using the survey design can improve the estimator of the *Gini coefficient*.

On the importance of measuring payout yield: implications for empirical asset pricing
J. Boudoukh et al.
Journal of finance (2007)

Try replacing dividend yield by *payout* (dividends plus repurchases) or net payout (dividends plus repurchases minus issuances) yields.

The impact of constraints on value-added A non-cynical approach
B. Scherer and X. Xu
Journal of portfolio management (2007)

Confusing article trying to explain how to use Lagrange multipliers to measure the impact of constraints on the information ratio (the authors have apparently never heard about “matrices”, they forget to define the quantities in their formulas, they do not understand the

difference between “term” (an element of a sum) and “factor” (an element of a product), etc.).

Alternative metrics
S. Zaker

Journal of alternative investments (2007)

Empty article reminding us to provide *confidence intervals* whenever we compute information ratios (IR).

Building a hedge fund portfolio with kurtosis and skewness
M. Anson et al.

Journal of alternative investments (2007)

Polynomial goal programming (PGP) solves multi-objective optimization problems as follows:

- Solve the problem for each objective separately; let f_i^{Max} be the optimum of the i th objective f_i ;
- Minimize $\prod (1 + (f_i^{\text{Max}} - f_i))^{\alpha_i}$ under the same constraints; the α_i are user-chosen parameters.

The authors prefer PGP to a linear combination of the objectives – the methods are actually equivalent, under a monotonic transformation of the objective functions.

An analysis of trade-size clustering and its relation to stealth trading
G.J. Alexander and M.A. Peterson
Journal of financial econometrics (2007)

Stealth traders tend to use medium-sized *rounded* (500, 1000, 5000) transaction in an attempt to disguise their trades.

Optimizing benchmark-based portfolios with hedge funds
I. Popova et al.

Journal of alternative investments (2007)

To maximize the expected utility, the probability of outperforming a benchmark, possibly corrected with the expected shortfall, in a non-gaussian set-up:

- generate N scenarios; the authors model hedge fund returns as a mixture of two gaussians, designed to fit the first four moments, corrected for various biases (survivorship, autocorrelation, etc.);
- optimize; this gives an upper bound on the true optimal value;
- estimate the expected utility of the resulting weights on a new random sample of N scenarios; you get a lower bound;
- if the confidence interval given by those bounds is too small, increase N .

Shrinking the covariance matrix
D.J. Disatnik and S. Benninga
Journal of portfolio management (2007)

Comparison of several covariance matrix estimators (sample variance, shrinkage with an optimized shrinkage coefficient, equal-weighted portfolio of variance estimators, diagonal matrix): the shrinkage estimators

give better risk forecasts for the unconstrained minimum variance portfolio but no estimator leads to a good forecast of the standard deviation of the constrained (long-only) minimum variance portfolio.

***How do performance measures perform?
Examining precision and stability***

G. Hübner

Journal of portfolio management (2007)

Performance or risk measures such as *beta*, *Jensen's alpha* (the intercept of the regression of the returns against the market returns), *Treynor ratio*

$$GTR_i = \alpha_i B_{\text{market}} R / B_i R$$

where $r_i = \alpha_i + B_i R + \varepsilon_i$, *information ratio* (IR), *Sortino ratio*, Modigliani's M^2 (not defined), Ω can be compared with:

- the rank correlation;
- the *concordance correlation coefficient*

$$\rho = \frac{2\sigma_{XY}}{\sigma_X^2 + \sigma_Y^2 + (\mu_X - \mu_Y)^2};$$

- Cohen's Kappa, after binning the stocks into winner and losers for each measure (it estimates the number of matching pairs not due to chance),

$$\kappa = 2 \frac{WW + LL}{N} - 1.$$

The author prefers the Treynor ratio.

***Where do alphas come from?
A new measure of the value of active
investment management***

A.W. Lo (2007)

Performance is traditionnally measured by ratios,

$$\text{Sharpe ratio} = \frac{E[\text{excess returns}]}{\sigma(\text{excess returns})}$$

$$\text{Treynor ratio} = \frac{E[\text{excess returns}]}{\beta}$$

$$\text{Information ratio} = \frac{\alpha}{\sigma(\text{noise})}$$

where

$$\text{excess returns} = \alpha + \beta \cdot (\text{market excess returns}) + \text{noise}$$

(for our market-neutral strategies, $\beta = 0$, so the Sharpe ratio and the information coefficient coincide).

The author suggests a decomposition of the expected portfolio returns into a passive and a non-passive com-

ponents:

$$\begin{aligned} E[\text{excess returns}] &= E \left[\sum_i w_i r_i \right] \\ &= \sum_i E[w_i r_i] \\ &= \sum_i (\text{Cov}(w_i, r_i) + E[w_i]E[r_i]) \\ &= \sum_i \text{Cov}(w_i, r_i) + \sum_i E[w_i]E[r_i] \\ \text{Active component} &= \sum_i \text{Cov}(w_i, r_i) \\ \text{Passive component} &= \sum_i E[w_i]E[r_i] \end{aligned}$$

$$\text{Active ratio} = \frac{\text{active component}}{\text{active component} + \text{passive component}}$$

***Exchange rates and the conversion of
currency-specific risk premia***
A. Eisenberg and M. Rudolf
European financial management (2007)

In an economy without arbitrage oportunities, there is a positive *stochastic discount factor* (SDF) that prices all assets

$$P_{i,t} = E_t \left[\frac{M_T}{M_t} X_{i,T} \right]$$

where $X_{i,T}$ is the payoff of asset i and M_T/M_t is the SDF.

In an international setup, there is one SDF for each currency. In a complete market, the exchange rate can conveert stochastic discount factors accross currencies (in incomplete markets, the SDF is not unique).

Hedging by sequential regressions revisited
A. Černý and J. Kallsen (2007)

One can compute a hedging strategy in an incomplete market in a sequence of 1-period least squares regressions backward in time.

The promise and peril of real options
A. Damodaran

Standard discounted cash flow models assume that the cash flows are deterministic; they are actually probabilistic and even contain *embedded options*, which have to be taken into account to properly value a cash flow or any corporate decision.

Examples include: the option to delay, expand, abandon a project, patents, natural ressources, etc.

The article recalls what an option is.

***Investment under uncertainty
and volatility estimation risk***
G. Dotsis et al.

Real option pricing requires an estimate of the volatility from very few data points, but practitioners often neglect to compute the resulting confidence interval: it can be large, and asymmetric.

***Evolution analysis of large-scale software
systems using design structure matrix
and design rule theory***
M.J. LaMantia et al.

Modular software is good: the option (as in “option pricing” – these are *real options*) to replace a component makes it more valuable – a counter-example being the “complexity disaster” Windows Vista.

The *design structure matrix* (DSM) represents dependencies between the modules; once spotted, circular dependencies can be removed by adding one more module.

***The effect of shortfall as a risk measure for
portfolios with hedge funds***
A. Lucas and a. Siegmann

The choice of the downside risk measure used to optimize a portfolio of hedge funds is important: the expected shortfall can lower small losses while increasing the risk of large crashes; quadratic shortfall and semi-variance do not have this problem.

***Variance, volatility swaps
and hedging your equity portfolio***
S.E. Satchell
Journal of asset management (2007)
(editorial)

Volatility and variance swaps (whose valuation and hedging are model-independent in a jump-free world) seem to provide a hedge against market collapse – as did *portfolio insurance* until October 1987.

***Volatility filter for index tracking and
long-short market-neutral strategies***
J. Miao
Journal of asset management (2007)

Only rebalance your portfolio when the volatility regime changes.

Portfolio size and diversification
L.R. Irala and P. Patil (2007)

Portfolio diversification benefits become negligible beyond 15 stocks – other authors claim it never becomes negligible.

***Nonparametric estimation of state-price
densities implicit in financial asset prices***
Y. Ait-Sahalia and A.W. Lo
Journal of finance (1998)

The *state price density* (SPD, *i.e.*, the density of the prices of Arrow–Debreu securities, often called *risk-neutral density*) can be computed:

- parametrically, using a model of the underlying asset price dynamics (e.g., a geometric brownian motion for the Black–Scholes model);
- parametrically, by imposing the family of the distribution and using actual option prices;
- in a bayesian way (as above, but with a prior).

The authors present a non-parametric estimator of the risk-neutral density using a *kernel regression* (sometimes called *local regression*).

***Effects of stochastic interest rates and
volatility on contingent claims***
N. Kunitomo and Y.-J. Kim (2005)

Black-Scholes option pricing makes three assumptions: the log-returns of the underlying are gaussian, the volatility is constant, the interest rate is constant. Generalizations often only relax one of those assumptions: the authors relax two of them.

***Option valuation with conditional
heteroskedasticity and non-normality***
P. Christoffersen et al. (2006)

Discrete risk-neutral valuation.

***The shape and term structure of the index
option smirk: why multifactor stochastic
volatility models work so well***
P. Christoffersen et al. (2007)

Prefer 2-factor stochastic volatility models:

$$dS = rSdt + \sqrt{V_1}Sdz_1 + \sqrt{V_2}Sdz_2$$

$$dV_1 = (a_1 - b_1V_1)dt + \sigma_1\sqrt{V_1}dz_3$$

$$dV_2 = (a_2 - b_2V_2)dt + \sigma_2\sqrt{V_2}dz_4$$

***Options prices under arithmetic brownian
motion and their implication for
modern derivatives pricing***
Q. Liu (2007)

Risk-neutral option pricing under *arithmetic* brownian motion (unrealistic, but a good interview question) violates the no-arbitrage principle.

***The risk and return characteristics of the
buy-write strategy on the Russel 2000 index***
N. Kapadia and E. Szado
Journal of alternative investments (2007)

The *buy-write strategy* (write a call and buy the underlying) on the S&P 500 beats the S&P 500 on a risk-adjusted basis (lower returns but much lower volatility); for the Russel 2000, it still works with 1-month at-the-money options.

Commonality in the time-variation of stock-stock and stock-bond return comovements

R.A. Connolly et al.

Journal of financial markets (2007)

Implied volatility (of equity index options) can help predict the link between stock and bond returns.

A reduced rank regression approach to coincident and leading indexes building

G. Cubadda (2005)

To monitor the business cycle, one can build a *coincident index* (that changes when a recession occurs) and a *leading index* (that changes before a recession starts).

The article uses *reduced rank regression* (a vector autoregressive (VAR) model with a reduced rank restriction) to build those indices from a set of cointegrated macroeconomic variables.

Do losses linger?

R. Garvey et al.

Journal of portfolio management (2007)

Traders who experienced a loss in the morning are more risk-seeking in the afternoon: this *disposition effect* is in agreement with *prospect theory*, defined here as the maximization of an *s-shaped* “utility”, function of gains and losses instead of total wealth.

Expected returns and Markov-switching illiquidity

T.R. Henry and J.T. Scruggs (2007)

When the markets are liquid, the prices are driven by fundamentals; when the markets become illiquid, trading volume has a greater impact on prices. The authors use a 2-state Markov switching model for illiquidity, returns and volatility to quantify this *illiquidity premium*.

The market P/E ratio, earnings trends and stock return forecasts

R.A. Weigand and R. Irons

Journal of portfolio management (2007)

The relation between P/E and future returns is linear when $P/E \geq 20$, but not beyond (in the US).

Coming to portfolios near you: investment ideas you should be paying more attention to

S. Welch

Journal of wealth management (2007)

“Postmodern portfolio theory” uses:

- a downside risk measure;
- minimum acceptable returns (MAR);
- non-gaussian returns.

The article also suggests to separate the strategy into *core* (passive) and *satellite* (absolute returns).

Tilted nonparametric estimation of volatility functions

P.C.B. Phillips and K.-L. Xu (2007)

In a heteroskedastic regression

$$Y = m(X) + \sigma(X)\varepsilon$$

or in a non-linear time-series model

$$Y_i = m(Y_{i-1}) + \sigma(Y_{i-1})\varepsilon_i$$

one can fit the variance σ^2 by local polynomials, à la Nadaraya–Watson.

Global and regional sources of risk in equity markets: evidence from factor models with time-varying conditional skewness

A.R. Hashmi and A.S. Tay (2004)

The GARCH model can be generalized with modified Student innovations (to account for skewness) and auto-regressive skewness.

Market timing with aggregate and idiosyncratic stock volatilities

H. Guo and J. Higbee

Journal of portfolio management (2007)

The model

$$\text{market returns} \sim \text{market variance}$$

is not significant, but if you add idiosyncratic volatility, it is. This can be used to time the market.

Optimal gearing: not all long-short portfolios are efficient

S. Johnson et al.

Journal of portfolio management (2007)

Confusing article: the authors, who worship Grinold and Kahn (for the latter, this is even self-worshipping) and do not understand the difference between = and \implies , try to prevent the reader from making even worse confusions.

We don't quite know what we are talking about when we are talking about volatility

D. Goldstein and N.N. Taleb

Journal of portfolio management (2007)

Warning about the confusion between mean absolute deviation (MAD) and standard deviation.

***Implied volatility from Asian options
via Monte Carlo methods***
C.-O. Ewald et al.

Many people calibrate the volatility surface using European options and then use it with Asian ones (e.g., to compute their vega), oblivious to the dependence of the volatility surface on the type of option.

The article uses the “logarithm trick”

$$\frac{\partial E[h(X_\sigma)]}{\partial \sigma} = E \left[h(X_\sigma) \frac{\log f(X_\sigma, \sigma)}{d\sigma} \right]$$

(where $f(\cdot, \sigma)$ is the density of the law of X_σ)

to express the vega of an Asian option as a weighted expectation of $h(X_\sigma)$, which can then be computed via Monte Carlo methods.

***Optimizing the retirement portfolio: asset
allocation, annuitization and risk aversion***
W.J. Horneff (2006)

Investment advice for retirement is either *life annuity* (i.e., the payment of the same sum every year until death) or *phased withdrawal plan* (a normal portfolio, say 60% stock, 40% bond, from which we take a fixed proportion for consumption every year – the proportion can also be chosen as $1/T$, where T is the maximum or expected number of remaining years of life).

The article suggests a blend of the two. They do not seem to mention *portfolio insurance*.

***A multivariate commodity analysis and
applications to risk management***
R.H. Börger et al.

Asset (here, commodity futures) returns are not gaussian and can be modeled by a *generalized hyperbolic distribution*, the mixture of a gaussian and a *generalized inverse gaussian*:

$$X = \mu + W\gamma + \sqrt{W}AZ$$

where $Z \sim N(0_k, I_k)$, $W \sim GIG(\lambda, \chi, \psi)$, $\mu, \lambda \in \mathbf{R}^d$, $Z, A \in \mathbf{R}^{d \times k}$.

This gives better value-at-risk and expected shortfall estimations.

***The m out of n provisional call: an auxiliary
reversed binomial tree approximation***
Q. Liu (2006)

A *convertible bond* (CB) is a bond, with an option to transform it into a stock.

A *hard call* is a CB which may be exercised anytime after (say) the first two years.

A *soft-call* is a CB which may only be exercised if the underlying stock closes above a prespecified trigger m out of n consecutive days.

Even though this is a path-dependent option, it can be priced with a binomial “tree” (a lattice, really) without keeping all the path histories in memory.

Composing contracts:

an adventure in financial engineering
S. Peyton Jones et al.
Functional Pearl, ICFP (2000)

Haskell (or any other functional language) can be used to *describe* complicated financial contracts (options on options on...on options, with complicated cash flows) – the industry lack such a precise notation – and then to price them (but there are still optimizations to be made; and one might prefer to use C for the final computations).

***Constructive homological algebra
and applications***
J. Rubio and F. Sergeraert (2006)

Another unlikely application of functional programming.

Random walks and electric networks
P.G. Doyle and J.L. Snell (1984)
arXiv:math/0001057

Around Polya’s theorem, claiming that random walks return to the origin in dimension 2 but not beyond – in other words, wind instruments are possible in our 3-dimensional world but not in Flatland.

***The electrical resistance of a graph captures its
commute and cover time*** A.K. Chandra et al.
(Article on the same subject.)

***Random matrix theory and robust covariance
matrix estimation for financial data***
G. Frahm and U. Jaekel
arXiv:physics/0503007

The authors “develop” (physicists tend to forget to define the quantities the use) a robust M -estimator of the variance matrix, the *spectral estimator* for which the largest eigenvalue follows the Marčenko–Pastur distribution, even for a heavy-tailed elliptic random vector.

***Stochastic programming models for asset
liability management***
R. Kouwenberg and S.A. Zenios (2001)
in *Handbook of asset and liability management*

Scenario trees for a stochastic program, in an asset liability management (ALM) context, can be generated with one of the following methods:

- random sampling from the model (you might want to trim down the resulting tree, as in *importance sampling*);
- adjusted random sampling (use *antithetic sampling to fit every odd moment of the underlying distribution*);
- produce returns that match the first few moments of the distribution (this optimization problem can be longer to solve than the final ALM problem).

You should make sure not to introduce arbitrage opportunities because of discretization or approximation

errors: the no-arbitrage condition can be added as a linear constraint.

Stochastic programming needs model generation tools – in particular, there is still no stochastic optimization language.

ALM can also be tackled with mean-variance optimization: just add a “liability” asset with a prescribed weight.

This chapter also defines a *myopic* investor: a multi-period investor who behaves as a 1-period investor, for instance because of his constant relative risk aversion (CRRA), if he has a power utility.

***Extending algebraic modelling languages
for stochastic programming***
P. Valente et al.

It is easier to formulate optimization problems using a declarative language (AMPL, GAMS, AIMMS, MPL, to name a few) than with a list of huge matrices.

This article advocates the use of a similar language, SAMPL, for *stochastic programs*. It also reviews the various types of stochastic programs and gives a few examples (in particular, how to turn a multistage stochastic program into a deterministic one: expand all the scenarios and add non-anticipatory constraints).

SAMPL is implemented in SPInE (commercial).

Also check SMPS (not an algebraic language).

Efficient tests of stock return predictability
J.Y. Campbell et al.
Journal of financial economics (2006)

When performing statistical tests, do not overlook the effects of *autocorrelation*.

***Optimal continuous-time hedging
with leptokurtic returns***
A. Černý (2005)

When the market is incomplete, perfect hedging is impossible and one can resort to utility-based (dynamic) mean-variance hedging.

***Portfolio optimization with stochastic
dominance constraints***
D. Dentcheva and A. Ruszczyński (2003)

In a scenario-based optimization, stochastic dominance constraints are straightforward to express.

(*missing title*)

The ARMA model can be generalized into a non-linear ARMA (NARMA)

$$y_{n+1} = F(y_n, \dots, y_{n-k}, \varepsilon_n, \dots, \varepsilon_{n-\ell}) + \varepsilon_{n+1}$$

or even a non-linear ARMA with exogenous variables (NARMAX) model

$$y_{n+1} = F(y_n, \dots, y_{n-k}, \varepsilon_n, \dots, \varepsilon_{n-\ell}, x_n, \dots, x_{n-k}) + \varepsilon_{n+1}$$

Universal portfolios
T.M. Cover
Mathematical finance (1991)

Given the history of total returns of a set of stocks x_{tj} , one can compute, a posteriori, the best constant-weight (rebalanced) strategy and try to beat it or at least to attain the same asymptotic rate of growth, without peering into the future but by allowing for changing weights.

The author suggests, at each time k , to average all the possible constant-weight strategies \mathbf{b} , weighing them by their past performance $S_k(\mathbf{b})$:

$$b(1) = \left(\frac{1}{m}, \dots, \frac{1}{m} \right)$$

$$b(k+1) = \frac{\int_B \mathbf{b} S_k(\mathbf{b}) d\mathbf{b}}{\int_B S_k(\mathbf{b}) d\mathbf{b}}$$

$$S_k(\mathbf{b}) = \prod_{t=1}^k \mathbf{b}' \mathbf{x}_t$$

where $B = \{ \mathbf{b} \in \mathbf{R}^m : \mathbf{b} \geq 0, \mathbf{b}' \mathbf{1} = 1 \}$ is the set of long-only fully-invested strategies.

This works, simply because the wealth of the strategy is the average wealth of the experts, all of which have an exponential growth rate, and an average of exponentials has (under suitable assumptions) the same asymptotic growth rate as the maximum. Buy-and-hold strategies are beaten because they are not diversified enough.

For $m = 2$ stocks, the computations are easy. Using

$$\int_{\substack{b_1, b_2 \geq 0 \\ b_1 + b_2 = 1}} f(b_1, b_2) db = \int_0^1 f(u, 1-u) du$$

$$B(1+x, 1+y) = \int_0^1 u^x (1-u)^y du$$

one gets

$$b_1(n) = \frac{\sum_{P_1 \sqcup P_2 = \llbracket 1, n \rrbracket} \prod_{t \in P_1} x_{t1} \prod_{t \in P_2} x_{t2} B(2 + |P_1|, 1 + |P_2|)}{\sum_{P_1 \sqcup P_2 = \llbracket 1, n \rrbracket} \prod_{t \in P_1} x_{t1} \prod_{t \in P_2} x_{t2} B(1 + |P_1|, 1 + |P_2|)}$$

where $P_1 \sqcup P_2 = \llbracket 1, n \rrbracket$ are the partitions of $\llbracket 1, n \rrbracket$ in two parts. The beta function is related to the binomial coefficients and can be computed in the same way.

In dimension m , the computations are similar, with a “multivariate beta function” whose computations require an m -dimensional array. Since the sums have m^n terms, the complexity is $O(m^n + n^m)$. Last, but not least, the important terms in those huge sums are very close to one: numerical instability prevents us from doing the computations in this way. (Since these are high-dimensional integrals, monte Carlo methods (importance sampling) might be useful.)

There are no assumptions for this strategy (this is a worst-case scenario), but the constant-weight benchmark is only optimal if there is no time-dependence in the data, *i.e.*, if the data are iid.

This is an expanding window strategy: with enough data, the weights will hardly move – this is not a dynamic strategy.

Efficient algorithms for universal portfolios
A. Kalai and S. Vempala
Journal of machine learning research (2002)

The definition of Cover's universal portfolio is exactly the Monte Carlo Markov Chain (MCMC) setup:

$$w_{\text{universal}} = \frac{\int w \text{Performance}(w) dw}{\int \text{Performance}(w) dw}.$$

To prove the convergence of the chain, the authors do not integrate over the simplex of all portfolios but limit themselves to those all of whose weights are at least δ_0 and they slightly modify the performance function.

Nonparametric kernel-based sequential investment strategies
L. Györfi et al. (2005)

In contrast with Cover's approach, the authors consider a stationary and ergodic market: there can be time-dependency; their benchmark is the *log-optimal portfolio*

$$b_n = \text{Argmax} E[\log(b'_n x_n) | x_{-\infty \rightarrow n-1}].$$

A *universal strategy* is a strategy that attains the same (asymptotic) rate of return without knowing the distribution.

The *histogram-based strategy* is a performance-weighted average of elementary strategies or *experts*; an expert is the best strategy, over windows of size k , according to the past history, corresponding to a discretization ℓ of the past returns. They consider all possible window sizes and the discretizations are defined from nested partitions of the space of returns, so that the size of the cells uniformly converge to zero.

(This is not implementable: most of the cells would be empty, unless you have a humongous amount of data; it will converge very slowly.)

The *kernel-based strategy* replaces the rigid partitions by a moving window:

$$b^{(k,\ell)}(x_{1 \rightarrow n-1}, s) = \text{Argmax}_{\substack{b \geq 0 \\ b'1=1}} \prod_{\substack{k < i < n \\ \|x_{i-k \rightarrow i-1} - s\| \leq c/\ell}} b' x_i$$

where $x_{1 \rightarrow n-1}$ is the whole history and $s = x_{n-k \rightarrow n-1}$ is the recent history.

The *general kernel-based strategy* is

$$b^{(k,\ell)}(x_{1 \rightarrow n-1}, s) = \text{Argmax}_{\substack{b \geq 0 \\ b'1=1}} \frac{\sum_i w_i^{(k,\ell)} \log b' x_i}{\sum_i w_i^{(k,\ell)}} \\ w_i^{(k,\ell)} = K_k(\ell \cdot (x_{i-k \rightarrow i-1} - s))$$

where K_k is a kernel (on \mathbf{R}^{kd} , where d is the number of assets), such as $K_k = \mathbf{1}_{\|x\| \leq c}$.

This is an expanding window strategy; the stationarity assumption makes it non-robust to structural changes.

Universal portfolios with side information
T.M. Cover
IEEE transactions on information theory (1996)

In the definition of Cover's universal portfolio,

$$w_{\text{universal}} = \frac{\int w \text{Performance}(w) dw}{\int \text{Performance}(w) dw}$$

one can replace the uniform distribution dw by a Dirichlet($\frac{1}{2}, \dots, \frac{1}{2}$) prior – the worst-case performance is slightly better and, this being (bayesian) *conjugate* distribution, the computations are marginally faster.

Uniform or Dirichlet universal portfolios attain their worst-case performance for *Kelly market sequences*, *i.e.*, when, in each period, all stocks but one go bankrupt (similar to horse races, where a single horse wins).

This article also explains how to use side information: the benchmark no longer has constant weights but the weights are the function

$$\text{side information} \mapsto \text{weights}$$

that maximizes the total returns on the period. The corresponding universal weights are

$$w_{\text{universal}}(y) = \frac{\int w \text{Performance}(w|y) dw}{\int \text{Performance}(w|y) dw}$$

where y is the side information.

Universal portfolios can also be used to combine experts: if the experts have constant weights, just use them instead of all the constant reweighted strategies (the integrals become sums); even better, consider them as new assets, add them to the universe and compute the universal portfolio of this enlarged universe.

The article ends with some details about the exact computation of those portfolios, in the Dirichlet case – this is still exponential.

Gestion de portefeuille et croissance optimale
D. Herlemont (2004)
<http://yats.com/>

The *Kelly criterion* invests the same fraction of your wealth in a game (e.g., a coin-tossing game where you receive twice the value of your bet when you win); the fraction is chosen so as

- to maximize the long-term growth rate $\frac{1}{t} \log \frac{W_t}{W_0}$;
- to maximize the expectation of the logarithmic utility $u(W) = \log W$ (log-optimal portfolios are sometimes called *Kelly portfolios*);
- to maximize the median wealth;
- to minimize the expected time to reach a given wealth;
- to minimize the entropy (?).

The Kelly criterion can be generalized to any binomial setting (you earn α on tails and β on heads, as in *bracket tracking*, *i.e.*, profit taking and stop-loss rules) or for any distribution of returns.

Here are some more generalizations of the Kelly strategy:

- use power utility: this is the *fractional Kelly criterion*;
- impose a constraint on the maximum acceptable drawdown and only reinvest a constant fraction of the wealth in excess of the drawdown;
- maximize the growth rate of the utility:

$$\liminf \frac{1}{t} E[\text{utility } W(t)];$$

- add transaction costs – but the non-trading intervals can be very large and the rebalancing frequency very low (several years);

The Kelly criterion might still work if the asset has a negative trend: it only requires that $E[\text{returns}] > 0$, while a positive trend is the more restrictive condition $E[\log \text{returns}] > 0$; the Kelly criterion is a constantly-rebalanced strategy while the asset alone is a buy-and-hold strategy.

The drawdown of a Kelly strategy follows a power law with exponent 2 (under a gaussian assumption); even worse, assuming gaussian returns when they are skewed or have fat tails leads to an excessive leverage and larger losses.

The Kelly strategy is sometimes referred to as a *volatility pump*: for a log-normal process without drift, the optimal growth is proportional to the variance. D. Farmer likens this phenomena to tapping energy from waves or tides.

Cover's *universal portfolios* are a non-parametric analogue of the Kelly criterion:

- Cover averages the constant reweighted strategies, weighing them with their performance;
- Györfi et al. average experts, *i.e.*, non-parametric functions that use the returns in the past k periods to produce suggested weights;
- One could also average statistical experts, *i.e.*, parametric statistical models (regression, GARCH, etc.) that map previous returns to weights.

This article (mentions but does not) present a generalization of Györfi et al.'s non-parametric experts without a long-only constraint.

Optimal gambling systems for favorable games **L. Breiman (1961)**

In a favorable game, the *Kelly principle* bets each time the same fraction of the current wealth, chose so as to maximize the rate of growth or, equivalently, the expected logarithmic utility.

A new interpretation of information rate **J.L. Kelly (1956)**

Kelly's original, information-theoretic article.

Can we learn to predict the best stock?

A. Borodin et al.
Journal of artificial intelligence research (2004)

The authors add a new strategy to the list of portfolio selection algorithms:

- *exponentiated gradient*: greedily choose the best portfolio for yesterday's market with a penalty for moving far from yesterday's portfolio;
- Cover's universal portfolio:

$$b(k+1) = \frac{\int_B \mathbf{b} \text{Performance}_k(\mathbf{b}) d\mathbf{b}}{\int_B \text{Performance}_k(\mathbf{b}) d\mathbf{b}}$$

- where $d\mathbf{b}$ is the prior distribution on the weights, for instance a uniform or Dirichlet distribution;
- algorithms that predict the best stock, such as the prediction component of the Lempel-Ziv compression algorithm.

Choose a window size w , consider the stock log-returns in the previous two non-overlapping windows x_{-1} and x_{-2} and compute the cross-correlation matrix $\text{Cor}(x_{-2}, x_{-1})$. If stock i outperformed stock j in the most recent period and if the log-returns of i in period -2 are correlated with those of j in period -1 , shift your investment from stock i to stock j by a proportion

$$\begin{aligned} \text{claim}_{i \rightarrow j} &= 0 \quad \text{unless } \mu_{-1}(i) > \mu_{-1}(j) \\ &\quad \text{and } \text{Cor}(x_{-2}, x_{-1})(i, j) \\ \text{claim}_{i \rightarrow j} &= \text{Cor}(x_{-2}, x_{-1})(i, j) > 0 \\ &\quad + |\text{Cor}(x_{-2}, x_{-1})(i, i) -| \\ &\quad + |\text{Cor}(x_{-2}, x_{-1})(j, j) -| \end{aligned}$$

The strategy is too sensitive to the choice of the window size w , This can be improved on:

- Consider an equal-weighted average of all those strategies for $w \in \llbracket 2, 50 \rrbracket$;
- Instead of applying the strategy on stocks $\text{ANTICOR}_w(x_1, \dots, x_n)$, apply it on strategies: $\text{ANTICOR}_w(\text{ANTICOR}_2, \dots, \text{ANTICOR}_{50})$.

I would also try to change “correlation” into “cointegration”.

On the competitive theory and practice of portfolio selection **A. Borodin et al.**

Proceedings of the 4th Latin American Symposium on Theoretical Informatics (2000)

After an information-theoretic review of the portfolio selection problem, the authors present several prediction-based algorithms:

- the *add-beta prediction rule*, used to predict the next element of a binary sequence

$$\hat{P}[X_{T+1}] = \frac{\beta + \#\{X_i = 0, 1 \leq i \leq T\}}{2\beta + T}$$

can be used to predict the best performing stock - it is competitive with Cover's universal portfolio, but not universal (bad worst-case performance) and the convergence is likely to be even slower, especially for large markets (for which a lot of information is discarded);

- Lempel-Ziv trading.

Transaction costs do not affect the theoretical, asymptotic results, but on real data, they are a real problem; you can sometimes have decent results by rebalancing every month instead of every day, though.

On-line portfolio selection using multiplicative updates

D.P. Helmbold et al.

Mathematical finance (1998)

One can greedily choose the best portfolio for yesterday's market with a penalty for moving far from yesterday's portfolio, according to the *entropy distance*. The exact optimization problem

$$\begin{aligned} & \text{Maximize}_{\mathbf{w}_{t+1}} \eta \log(\mathbf{w}'_{t+1} \mathbf{x}_t) - d(\mathbf{w}_{t+1}, \mathbf{w}_t) \\ d(\mathbf{u}, \mathbf{v}) &= \sum_i u_i \log \frac{u_i}{v_i} \end{aligned}$$

would be too long to solve at each step; instead, it is approximated as

$$w_{i,t+1} = \frac{w_{i,t} \exp \frac{\eta x_{i,t}}{\mathbf{w}'_t \mathbf{x}_t}}{\sum_j w_{j,t} \exp \frac{\eta x_{j,t}}{\mathbf{w}'_t \mathbf{x}_t}}$$

(start with an equal-weighted portfolio). This *exponentiated gradient* portfolio selection strategy is linear in time and space and seems to have better small-sample properties than Cover's.

Efficient universal portfolios for past-dependent target classes

J.E. Cross and A.R. Barron

Mathematical finance (2003)

Very technical article explaining how universal portfolios in continuous time, without any stochastic model (just a couple of regularity assumptions about the paths), has a formulaic expression computable in $O(m^2)$ where m is the number of stocks.

Multivariate realized stock market volatility

G.H. Bauer and K. Vorkink (2006)

High frequency data allows us to compute the *realized volatility matrix* without using a moving, overlapping window or a statistical model.

Many variance matrix estimators are plagued by the positive-definiteness condition, which is non-trivial to impose: the article suggests to estimate the matrix logarithm of the variance matrix instead – the *matrix exponential* of a real symmetric matrix is positive definite and every positive definite matrix can be uniquely obtained in this way.

Construction of multivariate copulas and the compatibility problem

C. Laforge

Examples of bivariate copulas abound but examples of multivariate copulas are rare, besides the *archimedian copulas*

$$C(x_1, \dots, x_n) = \phi^{-1}(\phi(x_1), \dots, \phi(x_n)).$$

The *copula product*

$$C_1 * C_2(x, y) = \int_0^1 \frac{\partial C_1(x, t)}{\partial t} \frac{\partial C_2(y, t)}{\partial t} dt$$

can be generalized to produce a multivariate copula from n bivariate copulas

$$C(x_1, \dots, x_n) = \int_0^1 \prod_{i=1}^n \frac{\partial C_i(x_i, t)}{\partial t} dt$$

or, in terms of copula densities,

$$c(x_1, \dots, x_n) = \frac{\partial^n C(x_1, \dots, x_n)}{\partial x_1 \cdots \partial x_n} = \int_0^1 \prod_{i=1}^n c_i(x_i, t) dt.$$

For instance, the Gumbel copulas

$$C_i(x, y) = xy e^{\theta_i \ln x \ln y}$$

give rise to an n -parameter copula

$$C(x_1, \dots, x_n) = \frac{\prod_{i=1}^n x_i (1 - \theta_i \ln x_i)}{1 - \sum_{i=1}^n \theta_i \ln x_i}.$$

Selecting copulas for risk management

E. Kole et al. (2006)

To choose a copula among gaussian, Student (another elliptic copula, with more tail dependence and the strange property that the zero-correlation Student copula is not independent) and Gumbel (an archimedian copula (in particular, it is \mathfrak{S}_n -invariant) with even more tail dependence), one can use the traditional

goodness-of-fit tests for distributions, Kolmogorov–Smirnov and Anderson–Darling (a copula is a distribution on the cube $[0, 1]^N$):

$$D_{KS}^{\max} = \max_t |F(\mathbf{x}_t) - F_0(\mathbf{x}_t)|$$

$$D_{KS}^{\text{avg}} = \int_{\mathbf{x}} |F(\mathbf{x}) - F_0(\mathbf{x})| dF_0(\mathbf{x})$$

$$D_{AD}^{\max} = \max_t \frac{|F(\mathbf{x}_t) - F_0(\mathbf{x}_t)|}{\sqrt{F_0(\mathbf{x}_t)(1 - F_0(\mathbf{x}_t))}}$$

$$D_{AD}^{\text{avg}} = \int_{\mathbf{x}} \frac{|F(\mathbf{x}) - F_0(\mathbf{x})|}{\sqrt{F_0(\mathbf{x})(1 - F_0(\mathbf{x}))}} dF_0(\mathbf{x})$$

(The *tail dependence* can be defined as an asymptotic expansion of the copula in $(0, 0)$ and $(1, 1)$.)

***Why use QuantLib?*
N.P. Firth (2004)**

After recalling what free or open source software is, the author reviews a few “open source” financial libraries: Premia (X, Inria, non-free, closed to external contributions), Financial numerical recipes (C++, GPL, weak project management), Martingale (C++ and Java, GPL, weak project management) and Quantlib (C++, BSD licence, more active mailing list but dominated by a single company).

***Hedging contingent claims with constrained portfolios and nonlinear wealth dynamics*
D. Ebmeier (2007)**

A *super-replication strategy* for a contingent claim is a strategy whose payoff dominates (*i.e.*, is almost surely greater than) the contingent claim. The initial wealth of this strategy is an upper bound on the price of the contingent claim (one can similarly get lower bounds) and can be used to *hedge* the contingent claim.

This article examines what happens when the super-replicating strategy has to satisfy some constraints (*e.g.*, short-selling constraints).

***Optimal replication of contingent claims under portfolio constraints*
M. Broadie et al.**

Article on the same subject.

***Extreme observations and non-normality in ARCH and GARCH*
R. Bali and H. Guirguis
International review of economics and finance (2007)**

Traditional GARCH estimators are extremely sensitive to outliers, which produce biased coefficients and fat-tailed residuals; this article explains how to correct for outliers.

***Predictive systems: living with imperfect predictors*
L. Pástor and R.F. Stambaugh (2006)**

When trying to predict future returns, do not forget the time dimension: also use the lagged returns and predictors.

See also: mixed data sampling (MIDAS).

***The information content in implied idiosyncratic volatility and the cross-section of stock returns: evidence from the option markets*
D. Diavatopoulos et al. (2007)**

Realized idiosyncratic risk (after removing market, size and book-to-market) is a bad predictor of future returns; option-implied idiosyncratic risk is better.

***A note on the mean-variance analysis of self-financing portfolios*
Z. Bai et al. (2006)**

Beware of optimization as a drop-in replacement for a statistical procedure: even with large samples, it tends to create biased *estimators*.

***You don't have to bother Newton for implied volatility*
M. Li (2006)**

Implied volatility can be seen as a *special function* (like sine, exponential, Bessel functions, etc.) and computed in the same way, *e.g.*, with rational approximation.

***Stock market return, order flow and financial market linkages*
J.M. Moberg and G. Sucarrat (2007)**

Order flow imbalance (OFI) still works in small markets.

***The similarity between mean-variance and mean-Gini: testing for equality of Gini correlations*
E. Schechtman et al.**

Here is one more risk measure one can use instead of the standard deviation in a portfolio optimization: the *Gini mean difference* (GMD) of a random variable X is

$$\text{GMD } X = E |X_1 - X_2|$$

where X_1 and X_2 are iid with the law of X .

The article uses

$$\text{Cov}(X, F_X(X))$$

instead; considers the (non-symmetric) Gini matrix $\text{Cov}(X_i, F_X(X_j))$ where X_i and X_j are the returns of assets i and j and F_X is the cumulative distribution of the pooled returns of all the assets; and derives conditions under which “mean Gini optimization” reduces to a quadratic program.

Random intersection graphs with tunable degree distribution and clustering
M. Deijfen and W. Kets (2007)

An *intersection graph* is a graph whose vertices are subsets of a set X , with an edge between two subsets whenever they intersect.

A *random intersection graph* can be generated as follows: start with the set of vertices, assign a random weight (following a law you can choose/tune) to each vertex, assign a subset to each vertex, whose size is proportional to the weight, consider the corresponding intersection graph.

This produces a wide range of random graphs.

Primer on using neural networks for forecasting market variables
S.A. Hamid (2004)

Neural networks, trained on futures and spot prices, are better predictors of realized volatility (standard deviation of the daily log-returns over the next two months) than implied volatility is.

The article also contains a literature review and a list of financial applications of neural networks.

The value of transaction cost forecasts: another source of alpha
A. Coppejans and A. Madhavan
Journal of investment management (2007)

Predicting transaction costs can increase your capacity.

Predicting hedge fund failure: a comparison of risk measures
H. Park (2005, 2007)

Downside risk is a better predictor of failure than standard deviation.

Hedge fund investment through piecewise linear regression and optimization
F. Pan and B. Zeng (2007)

The size of a hedge fund has a non-linear impact on its return; it can be investigated by a piecewise linear regression.

Timing ability in the focus market of hedge funds
Y. Chen
Journal of investment management (2007)

Hedge funds do time the market.

How hedge fund beat the market
C. French and D. Ko
Journal of investment management (2007)

Hedge funds do not time the market.

Hedge fund risk dynamics: implications for performance appraisal
N.P.B. Bollen and R.E. Whaley

Computing the “alpha” of a strategy (say, a hedge fund) by removing the linear effect of a set of standard risk factors (market, book-to-market, size, etc.) cannot account for non-linearities or structural breaks: this article uses *optimal change point regression* to account for the latter (for the former, just use non-linear risk factors, as below).

Detecting structural breaks and identifying risk factors in hedge fund returns: a bayesian approach
L. Meligkotsidou and I.D. Vrontos

Another article on the same subject.

Can hedge fund returns be replicated? the linear case
J. Hasanhodzic and A.W. Lo
Journal of investment management (2007)

Linear clones of hedge funds often have inferior performance, but they are transparent, scalable and cheaper.

Hedge fund replication strategies: implications for investors and regulators
W. Fung and D.A. Hsieh (2007)

Hedge fund performance can be replicated, in a linear way, from a small set of *primitive trading strategies* (PTS) such as (since these are tradable, they are sometimes called *asset-based style* (ABS) factors):

- Call option on the market (to replicate a market timer);
- Lookback straddles (to replicate trend followers, who buy at the low and sell at the high: a *lookback call option* allows the owner to buy at the lowest price over the life of the option);
- BAA corporate bonds minus 10-year treasuries (to replicate fixed income hedge funds);
- “The merger fund” (to replicate a merger arbitrage strategy – no details given);
- Swap spread (?), yield curve spread (?), mortgage spread (?);
- US stock market;
- Small minus large stocks (to replicate long-short equity strategies);
- IFC emerging market index;
- CSFB high yield bond index (to replicate distressed securities hedge funds).

Will hedge funds regress towards index-like products
W. Fung and D.A. Hsieh (2007)

Detailed and badly typeset version of the above.

Statistical properties of short term price trends in high frequency stock market data
P. Sieczka and J.A. Hołyst
arXiv:physics/0703208

High frequency data fails the *runs test* (unsurprising, given their long memory): with the amount of data, one can even study the *distribution* of the run lengths.

Return predictability, economic profits and model mis-specification: how important are the better-specified models?
Y. Han (2007)

Simulations show that having the “right” return prediction model does not allow you to beat a good old VAR (vector auto-regressive model) – the choice of variables is important, though.

Furthermore, return *sign* forecasts, (rather than return prediction) are more profitable.

This article is a pledge for simple models.

Implied correlation from VaR
J. Cotter and F. Longin (2007)

In a gaussian setup, one can compute the correlation of two variables X and Y from the value-at-risk (VaR) at level α of X , Y and $xX + yY$ (*i.e.*, a portfolio of X and Y with weights x and y): this is the *implied correlation* – it depends on the level α , the weights x and y (and also the frequency of the data).

The article examines the distribution of the implied correlation (this is just a family of estimators of the correlation) – beware of the bias.

An empirical study of multi-objective algorithms for stock ranking
Y.L. Becker et al.

Genetic algorithms can be used to simultaneously optimize several goals: put the solutions on separate “islands”, one for each goal, and have some migrate from time to time.

This is very similar to *data envelopment analysis* (DEA) but I do not expect the algorithm to converge towards a uniquely defined (or meaningful) solution; if the migration rate is well chosen, it should converge to a (set of) point(s) on the efficient frontier, not far away from the optimal solutions of the one-goal problems.

The algorithm is applied to a stock selection problem with the following goals: information ratio (IR), information coefficient (IC) and intra-fractile hit rate (IFHR), *i.e.*, proportion of stocks in the top (resp. bottom) decile that outperform (resp. underperform) the average.

Portfolio selection using evolutionary computational techniques

This (author-less) article suggests to replace each return estimate by three values (say, the median and

the quantiles) and optimizes three goals: maximize the middle value of the portfolio, minimize the distance between middle and low, maximize the distance between middle and high, subject to VaR-like constraints. The author is oblivious of the dependency between the stocks; his use of *fuzzy logic* looks bogus, his use of *genetic algorithms* is irrelevant (he combines the three goals into one).

Stable models for the distribution of equity capital
R. Fernholz
Intech (2004)

Stochastic portfolio theory can be used to model the evolution of the distribution of market capitalization over time.

Diversity and relative arbitrage in equity markets
R. Fernholz et al.
Intech (2004)

A market is *diverse* if the weight of its largest stock satisfies

$$\exists \delta > 0 \quad \forall t \in [0, T] \quad \mu_{(1)}(t) < 1 - \delta;$$

it is *weakly diverse* if

$$\exists \delta > 0 \quad \frac{1}{T} \int_0^T \mu_1(t) dt < 1 - \delta;$$

it is *asymptotically weakly diverse* if

$$\exists \delta > 0 \quad \lim_{T \rightarrow \infty} \frac{1}{T} \int_0^T \mu_1(t) dt < 1 - \delta.$$

In a weakly diverse market, there are arbitrage opportunities, e.g., invest

$$\frac{p}{w_1(0)^p} + (p - 1)$$

in the market and $-p$ in stock 1, where $p \gg 1$. The weights are so large that the strategy is not implementable; furthermore, there are market models, such as the *Atlas model* (defined in a later article by the same authors) that are almost diverse but with no arbitrage opportunity.

The article also investigates the consequences on long-term option pricing.

Relative arbitrage in volatility-stabilized markets
R. Fernholz and I. Karatzas
Intech (2004)

A more general condition than diversity for the existence of a relative arbitrage.

Stochastic portfolio theory: an overview
R. Fernolz and I. Karatzas
Intech (2006)

Clear and exhaustive review of stochastic portfolio theory.

Stocks as lotteries: the implications of probability weighting for security prices
N. Barberis and M. Huang (2007)

Prospect theory differs from expected (concave) utility theory:

- The utility function is concave over gains and convex over losses; it has a kink (*i.e.*, left and right derivatives are different) at the origin;
- The investor does not maximize the expected utility but a weighted utility: the probabilities are transformed (but these are not subjective probabilities, just decision weights: the investor is perfectly aware of the objective probabilities); as a result, low probabilities are overweight: the investor wants both lottery and insurance.

Prospect theory is incompatible with first-order dominance, but can be modified into *cumulative prospect theory*, which applies the weights to the cumulative distribution function. Often, one chooses

$$v(x) = \begin{cases} x^\alpha & \text{if } x \geq 0 \\ -\lambda(-x)^\alpha & \text{if } x < 0 \end{cases}$$

for the value function and

$$w(p) = \frac{p^\delta}{(p^\delta + (1-p)^\delta)^{1/\delta}}$$

for the weighting function. Psychological studies suggest $\alpha = 0.88$, $\lambda = 2.25$, $\delta = 0.65$.

Under gaussian assumptions (more generally, in the absence of skewness), cumulative prospect theory is consistent with the CAPM (capital asset pricing model); however, the weighting function creates mispricing for assets with skewed returns – but the authors are not convinced that it can be arbitrated away.

Cumulative prospect theory can explain that assets with a high idiosyncratic skewness, such as IPOs, private equity, distressed stocks, deep out-of-the-money options, are overpriced and earn a low average return – this leads to the *volatility smile*. It also explains why household portfolios lack diversification.

To test (and use) that positively skewed stocks earn lower average returns, one would need to forecast future skewness: past skewness does not work, but cross-sectional industry skewness does.

Probability elicitation, scoring rules and competition among forecasters
K.C. Lichtendahl and R.L. Winkler (2007)

Competitive forecasters tend to report more extreme probabilities.

A forecasting model for stock market diversity
F. Audrino et al.
Intech (2006)

To predict changes in market diversity and switch between a capitalization-weighted and a diversity- (or, better, equal-) weighted portfolio, one can use:

- an AR(1)-GARCH(1,1) model;
- a *generalized regime-switching* (GRS) model, *i.e.*, a mixture of AR(1)-GARCH(1,1)'s;
- a *generalized tree-structured* (GTS) model, *i.e.*, the analogue of a decision or regression tree whose leaves contain AR(1)-GARCH(1,1) models.

The predictive variables are:

- 1-month US treasury bill rate;
- 60-month zero-coupon bond rate;
- CPI;
- PPI of finished goods;
- Index of "help wanted" advertising in newspapers;
- Unemployment;
- Growth rate of industrial production;
- S&P 500 monthly log-returns;
- Large/small \times low-/mid-/high-value portfolio returns;
- Market returns minus 1-month treasury bill;
- SMB (large minus small) and HML (high minus low value) returns;
- Industry portfolios.

They find three regimes, that can be interpreted with the Fama–French factors.

The *diversity timing* strategy provides a "significant" 17bp increase in annual returns (after transaction costs) when compared with a monthly-rebalanced equal-weighted strategy.

Tree-structured GARCH models
F. Audrino and P. Bühlmann (2000)

The regression-tree idea can be applied to times series modeling: put GARCH(1,1) or AR(1)-GARCH(1,1) models in the leaves.

The authors use S-Plus but do not provide any code.

Estimating and predicting multivariate volatility thresholds in global stock markets
F. Audrino and F. Trojani (2003)

The tree-structured AR-GARCH model can be applied to national stock indices, the conditional variables being local and US market returns. There are usually three regimes: two driven by local information, one by US information.

Growth optimal investment strategy efficacy: an application on long run Australian equity data
B.F. Hunt
Investment management and financial innovations (2005)

The *growth optimal portfolio* is not diversified: it only contains 2 or 3 stocks. To increase diversity, one can regularize the sample variance matrix as in a ridge regression:

$$V = V_{\text{sample}} + \lambda I.$$

Incorporating estimation errors into portfolio selection: robust portfolio construction

S. Ceria and R.A. Stubbs
Journal of asset management (2006)

Robust optimization usually considers the worst-case scenario in ellipsoidal or box-shaped uncertainty regions, meaning that all the errors have a negative effect. This article suggests to consider the subset of those uncertainty sets where the errors cancel out.

Experiments in robust portfolio optimization
D. Bienstock (2007)

The *uncertainty sets* used in robust portfolio optimization are often boxes or ellipses (some add the further constraint that the deviations sum up to zero); this article presents two new shapes for the uncertainty shares: one using fractiles (just specify how bad the situation can be in each decile) and another using the *adversarial value-at-risk* (AVaR), that uses mixtures to model the higher correlation during crashes (I am not sure it is different from the VaR of a mixture).

The article also details how to efficiently implement those optimizations.

Equity-style timing: a multi-style rotation model for the Russel large-cap and small-cap growth and value style indexes

B.G. Arshanapalli
Journal of asset management (2007)

Market timing often uses logistic regression to switch between two market segments – beyond two, use *multinomial regression*.

Portfolio choice beyond the traditional approach
F. Peñaranda (2007)

Not very readable (if you do not already know what he talks about, you will not understand it) but correct and fairly complete survey of asset allocation, with all the formulas you may need. The topics include:

- the portfolio weights as a statistic, whose distribution can be studied (surprisingly, this was missing from Meucci's book);
- the difference between *posterior distribution* and *predictive distribution* (the posterior distribution is for the present, the predictive distribution is its projection to the investment horizon)
- *lower partial moments* (LPM) are linked to stochastic dominance (no details given) but are more computable and yield a complete order; mean-semi-variance optimization is a convex problem;

- generalizations of the Sharpe ratio: *Sortino ratio* (with the semi-variance), *Omega*, *rate of decay I* (this comes from *large deviation theory*)

$$P[\text{returns}_{0 \rightarrow T} \leq 0] \sim e^{-I \cdot T}$$

- (but the Sharpe ratio gives a ranking similar to most alternatives);
- log-utility is indifferent to the investment horizon;
- the *Black-Litterman* approach can be used to tilt a long-term strategy (strategic asset allocation or SAA) towards a short-term view (tactical asset allocation or TAA);
- dynamic asset allocation (with insufficient details) and the *intertemporal CAPM* (ICAPM).

The impact of EMU on the equity cost of capital
G.A. Hardouvelis et al.

Journal of international money and finance

European monetary integration can be modeled as a *time-weighted CAPM*

$$r_{it} = \lambda_{i,t-1} \beta_i^{\text{market}} r_t^{\text{market}} + (1 - \lambda_{i,t-1}) \beta_i^{\text{local}} r_t^{\text{local}}$$

$$\lambda_{i,t-1} = \exp - |\gamma'_i X_{i,t-1}|$$

where X is a set of predictive variables: country effects are becoming less important.

Performance and distress indicators of new public companies
N. Beneda

Journal of asset management (2007)

To choose among IPOs (initial public offerings), use Ohlson's *O ratio* (a measure of bankruptcy risk, apparently the result of a logistic regression, whose coefficients were set in stone 30 years ago), the market-to-book ratio and the market share of the underwriter.

Long-term economic relationships from cointegration maps
R. Vicente et al. arXiv:physics/0701062

Correlation is a cross-sectional distance, blind to dynamic phenomena: it can be replaced by a measure of cointegration.

The author suggests to estimate the cointegration from the model

$$a_1 x_{1,t} + a_2 x_{2,t} + b = \varepsilon_t$$

$$\varepsilon_{t+1} = \gamma \varepsilon_t + \eta_t$$

$$\langle \varepsilon_t \rangle = 0$$

$$\langle \varepsilon_t^2 \rangle = \sigma$$

$$0 \leq \gamma < 1$$

using bayesian computations and physicist's approximations:

$$\begin{aligned}\hat{\gamma} &= \text{Argmax} \log \int p(\gamma|\varepsilon)p(\varepsilon|x_1, x_2) d\varepsilon \\ p(\gamma|\varepsilon) &\propto 1_{[0,1]}(\gamma) \left[\sum_{t=1}^{T-1} (\varepsilon_{t+1} - \gamma\varepsilon_t)^2 \right]^{-\frac{T-2}{2}} \\ p(\varepsilon|x_1, x_2) &\propto \varepsilon_t - x_{1,t} \sin \hat{\theta} + x_{2,t} \cos \hat{\theta} \\ \hat{\theta} &= \frac{1}{2} \arctan \left(2 \frac{\langle x_1 x_2 \rangle}{\langle x_1^2 \rangle - \langle x_2^2 \rangle} \right)\end{aligned}$$

The estimator $\hat{\gamma}$ can be seen as a measure of dissimilarity and can be transformed (this can make the plots more or less readable):

$$d_\alpha(x_1, x_2) = \hat{\gamma}^\alpha.$$

The dissimilarity matrix is sometimes modified further (two times series are similar if they interact with the other time series in the same way – this idea is also used in *spectral clustering*):

$$D_{i,j} = \sqrt{\sum_{k=1}^N (d_\alpha(x_i, x_k) - d_\alpha(x_j, x_k))^2}.$$

Before plotting (this is the *heatmap*, common with bioarrays), the rows and columns of the dissimilarity matrix D can be reordered: a greedy algorithm looks for a permutation matrix P that minimizes

$$\mathcal{F}(P) = \text{tr}(PDP'W)$$

where the weight matrix W can be chosen as $W_{ij} = \exp(|i - j| \sigma)$ (neighbourhood weights) or $W = XX'$ for some vector X such that $X_i > X_j$ if $i > j$ (side-by-side weights): this is the SPIN algorithm.

Finally, one can apply a clustering algorithm such as SPC (*superparamagnetic clustering* – I have no idea what it means).

**Information dynamics
and origins of uncertainty**
P. Garbaczewski
arXiv:cond-mat/0703147

An uncertainty principle can be expressed in information-theoretic terms, between $S(\rho)$ and $S(\tilde{\rho})$ where ρ is a probability density, $-\ln \rho(x)$ the *surprise level*, $S(\rho) = \langle -\ln \rho \rangle$ the *Shanon entropy* and $\tilde{\rho} = \mathcal{F}\rho$ is the Fourier transform of ρ .

**Optimal execution of portfolio transactions:
a review**
E. Kochieva
Carisma (2007)

The market impact has a long-term component (information leakage) and a short-term one (liquidity-demand-related price jumps), which add to the background price dynamics.

Trade execution is a trade-off between *market impact* (which decreases with time) and *timing risk* (which increases with time); it can be evaluated with respect to a benchmark such as *pre-trade* price (the most often used), VWAP or post-trade price.

The *VWAP strategy* trades a fixed percentage of the market value in each period; the optimal trading rate can be estimated by a quadratic optimization (*efficient trading frontier*).

This static strategy can be replaced by a dynamic one, where the trading rate is a function of the price, or by a full-fledged *stochastic programming* approach.

Continuing work on optimal trade execution
D. Bienstock
Carisma (2007)

One can consider a market impact model of the form

$$p_{t+1}/p_t - 1 = \alpha * \text{size}^\pi$$

with a fixed (very small) α and π varying in $(0, 1)$ according to a Markov chain. The optimal trade execution strategy can be obtained by dynamic programming (or *approximate dynamic programming*). This strategy is not robust to a misspecification of the Markov transition probabilities – assuming that π is constant is preferable.

One can also assume that π is not chosen at random but by an adversary (under some constraints to keep it around a certain value $\bar{\pi}$; the corresponding optimal strategy is more robust.

One can also consider what happens when two traders compete.

Someone also suggested to try with a hidden Markov model (HMM), *i.e.*, with unknown transition probabilities.

**Design of an FX trading system using
adaptive reinforcement learning**
M.A.H. Dempster
Carisma (2007)

Recurrent reinforcement learning (RRL), *i.e.*, a recurrent 1-layer neural network, can be used to turn returns time series into a buy/sell/wait order, so as to maximize a *moving-average Sharpe ratio*,

$$\frac{\text{EWMA}(\text{returns})}{\text{EWMA}(\text{returns}^2)}.$$

This used to work, one decade ago. One can try to improve the model as follows:

- Do not only consider past returns: add in technical indicators – actually, this does not add anything: RRL already extracts all the relevant information;
- Replace the transaction costs parameter by something larger than the bid-ask spread, so as to favour trades with larger returns;
- Fix the instability in the neural network weights by shrinking them;

- Update the weights twice at each step, *i.e.*, replace $w_t = f(x_t, w_{t-1})$ with $w_t = f(x_t, f(x_t, w_{t-1}))$.
- Add a risk and performance management layer in the algorithm, with a stop-loss and a shutdown procedure (to stop investing when the algorithm becomes unstable after a regime switch); the suggested risk measure takes into account both total loss and the size of the individual losses (this is similar to Omega):

$$\Sigma = \frac{\sum r_i^2}{\sum r_{i+}^2};$$

- Update the neural network weights at each step, but do not update the meta-parameters (transaction cost, trading threshold, etc. – there are five of them) that often;

The following improvements have not been tested yet:

- Add other information, such as the order flow or the limit order book;
- Use the algorithm on several currency pairs; generalize the risk control accordingly;
- Apply the algorithm to market making instead of trading.

Optimal liquidation against a markovian limit order book

**P. Hewlett
Carisma (2007)**

This technical talk combined limit order book models (cf. D. Farmer's work – I attended one of talks last month) with dynamic programming to find the optimal way of executing a trade; some of the tools required are similar to those used to price American options (?); the robustness of the resulting strategy could be examined as in D. Bienstock's talk.

Automated new content and algorithmic trading **P. Gagner (RavenPack) Carisma (2007)**

They turn news into numbers using a Bayesian spam filter – unconvincing.

Portfolio optimization with drawdown constraints

**S. Uryasev
Carisma (2007)**

Maximum drawdown, average drawdown and conditional drawdown at risk (CDaR) can be expressed using piecewise linear functions and can therefore be optimized with a linear program (scenario-based) solver. The actual risk measure is not that important.

His company (AOrDa) sells an optimizer and he claims to run his own money with it, using sample returns as an alpha (and Monte Carlo simulations to estimate the risk); surprisingly, he is aware that this momentum strategy does not perform that well because of reversal effects.

Applying stochastic programs to improve investor performance

**J.M. Mulvey
Carisma (2007)**

The title sounded interesting, but the talk was mostly empty:

- do not forget to rebalance (equal weights are good);
- look for volatility (or diversity);
- do not forget transaction costs;
- do not forget to diversify your portfolio (using *overlays*, if needed): with enough diversity and volatility, a *momentum* strategy can be quite successful.

The stressed difference between *dynamic stochastic control* (discretization of the state space, *i.e.*, of the investor's decisions) and *multi-stage stochastic program* (scenario tree) was not clear. The author suggests to analyze the result of those optimizations to derive *rules* that can then be back-tested.

Theory of acceptability indices

**D. Madan
Carisma (2007)**

To compare and choose among strategies, one can use *scale-free acceptability indices*, such as the Sharpe ratio and its generalizations: these are functions $\alpha(X)$ of the distribution of the returns X of the strategy. An acceptability index is *coherent* if it satisfies the following properties:

- convexity: $\alpha(X) \geq x, \alpha(Y) \geq x \implies \forall \lambda \in (0, 1) \alpha(\lambda X + (1 - \lambda)Y) \geq x$
- monotonicity, *i.e.*, compatibility with strong dominance: if $X \geq Y$ almost surely, then $\alpha(X) \geq \alpha(Y)$;
- scale invariance: $\forall \lambda > 0 \quad \alpha(\lambda X) = \alpha(X)$
- continuity: if $|X_n| \leq 1$, $\alpha(X_n) \geq x$ and X_n converges to X in probability, then $\alpha(X) \geq x$.

We would also like:

- second order monotonicity;
- arbitrage consistency: if $X \geq 0$ almost surely, then $\alpha(X) = \infty$
- expectation consistency: if $E[X] < 0$, then $\alpha(X) = 0$; if $E[X] > 0$, then $\alpha(X) > 0$.

Here are a few examples:

- The Sharpe ratio:

$$SR(X) = \frac{E[X]}{Sd(X)} \text{ if } E[X] > 0;$$

it is not monotonic and not arbitrage consistent;

- The gain-loss ratio:

$$GLR(X) = \frac{E[X_+]}{E[X_-]} - 1 \text{ if } E[X] > 0;$$

it is coherent, but small and large losses are considered equally bad;

- RAROC = $\frac{E[X]}{VaR(X)}$

- The *tilt coefficient* is the highest risk aversion that will say “yes” to the strategy (for the exponential utility):

$$TC(X) = \inf\{\lambda \geq 0 : E[Xe^{-\lambda X}] < 0\}$$

- The expected shortfall can be turned into a scale-free acceptability measure:

$$AIT(X) = \frac{1}{\inf\{\lambda \in [0, 1] : ES_\lambda(X) \geq 0\}} - 1$$

where the expected shortfall is

$$ES_\lambda(X) = E[X | X \leq F_X^{-1}(\lambda)]$$

and F_X^{-1} is the quantile function of X .

Actually, coherent acceptability measures can be characterized as follows (this is the same notion of coherence and the same characterization as for the coherent indices of satisfactions, see A. Meucci’s book, section 5.6): generalize the expected shortfall to

$$u(X) = \int_0^1 \psi'(s) F_X^{-1}(s) ds$$

where $\psi : [0, 1] \rightarrow [0, 1]$

is concave, non-decreasing, with $\psi(0) = 0$ and $\psi(1) = 1$; it can be seen as a deformation to be applied to the quantile function F_X^{-1} before computing the expected shortfall, equivalently, ψ' can be seen as weights (for the expected shortfall, the weight is constant on $[0, \lambda]$ and zero on $[\lambda, 1]$). To get the index of satisfaction, we actually need an increasing continuous family $(\psi_x)_{x \geq 0}$ of such functions ψ , with $\psi_0(y) = y$ and $\psi_x \rightarrow \delta_0$ when $x \rightarrow \infty$. The corresponding index of satisfaction is then (this is the same formula as for the expected shortfall, with $x = 1/\lambda - 1$):

$$AIW(X) = \inf\{x \geq 0 : u_x(X) < 0\}$$

For instance (these can be interpreted in terms of order statistics):

- MinVaR: $\psi(y) = 1 - (1 - y)^{x+1}$
- MaxVaR: $\psi(y) = y^{1/(1+x)}$
- MaxMinVaR: $\psi(y) = (1 - (1 - y)^{x+1})^{1/(1+x)}$
- MinMaxVaR: $\psi(y) = 1 - (1 - y^{1/(1+x)})^{x+1}$

Those results were applied to options and hedge funds; for the latter, the influence of the kurtosis was decomposed into *peakedness* and *tailweightedness*.

Algorithmic trading of hedge funds

N. Christofides
Carisma (2007)

A complete trading platform can be built as follows:

- Gather the data (some variables as daily, others monthly)
- Fill in the data, using the EM algorithm (contrary to what the speaker seems to think, this is *not* a data imputation algorithm: using it as such would lead to biased estimators, especially for dispersion estimators)

- Build a risk model, using independent component analysis (ICA) (principal component analysis (PCA) fails to get rid of cross-kurtosis)
- Model the (daily) time-series behaviour of the independent components (more complicated than that of the initial assets), using *neural networks* (fitted with *bionomic* (?) algorithm) or *wavelets*
- Discretize the market model into a *state transition graph* (STG); merge similar vertices to keep their number under control (but you can still have millions of them); make sure there are no arbitrage opportunities (it suffices to check for 1-period opportunities from each vertex);
- Proceed with dynamic programming (the model is daily but your investment horizon is longer); you may want to use a *state space relaxation* technique to replace this single large problem into several smaller ones.

The speaker also mentioned *real options*, but I still have no idea what it is.

Long-short portfolio optimization under the mean-variance-CVaR framework

G. Mitra
Carisma (2007)

Current optimizers are not limited to the variance as a risk measure: they can use mean absolute deviation (MAD), semi-variance, lower partial moments, value at risk (VaR), conditional value at risk (CVaR, also called expected shortfall (ES) or tail VaR (TVaR)) – the presenter gives more details as to how this can be expressed as a linear problem, provided you add more variables and stick to *scenario-based optimization*; he also explains how to write the long-short, leverage and number-of-assets constraints.

You can also use several risk measures, such as the variance and the CVaR: the efficient frontier is then 2-dimensional.

Independent component analysis

A. Robinson (APT)
Carisma (2007)

Principal component analysis (PCA) models data as a linear mixture of gaussian variables; the principal components are linear combinations (more precisely, rotations) of the observed variables that maximize the variance.

Independant component analysis (ICA) models the data as a linear mixture of independant non-gaussian variables: the independent components are linear combinations of the observed variables that maximize some measure of non-gaussianity, such as *kurtosis* or *negentropy* – a couple of years ago, APT were extremely careful never to use the “ICA” words when explaining their product.

Similarly, *second-order blind identification* (SOBI) models the data as a linear mixture of autocorrelated

variables – one could generalize this by considering the volatility process.

The audience raised a few questions:

- how robust is it?
- can we combine ICA and SOBI?

D. Madan raised a few objections:

- the assumption of a linear mixture of non-gaussian variables is not valid for financial data: it would create dependence between the necks, while we also/mainly observe (and worry about) dependence in the tails;
- furthermore, the independent components we get are not independent.

***Alpha budgeting
cross-sectional dispersion decomposed***
W. Yu and Y.M. Sharaiha
Journal of asset management (2007)

Anova (analysis of variance) can be used to decompose cross-sectional variance.

Improving returns-based style analysis
Daniel Mostovoy
(Northfield, 2007)

Returns-based style analysis (RBSA) tries to replicate the returns of a fund, or at least to explain its risk (for variance decomposition or to use it in a (fund-of-funds) portfolio optimization), from a handful of assets or *spanning indices*, in a linear way. The naive *constrained linear regression* (constrained because the weights have to be positive) can be improved:

- you can *compute confidence* intervals on the weights;
- for hedge funds, you can allow negative weights, add a cash asset and constrain on the leverage;
- you can allow regime shifts by choosing the estimation period using *CUSUM* – they have a CUSUM video on their website;
- plot the residuals against time (everyone should already be doing that) to spot slow style changes and use exponential weights to get rid of them; alternatively, use a *Kalman particle filter* (because of the constraints, you have to use Monte Carlo Markov Chain (MCMC) simulations)
- Correct for heteroskedasticity, lest the more volatile periods be counted more heavily: define the dispersion between the spanning indices, at a given date, as their (cross-sectional) average absolute returns difference; weigh the observations with the inverse of the square root of the dispersion;
- Be sure to include volatility-base indices; you can choose the spanning indices by *forward stepwise regression*;
- the proxy portfolio will be too diversified: replace it by its constituents and ask the optimizer to reduce the number of assets;
- You still cannot capture all the stock-specific risk: this can be fixed by adding cash and changing the

leverage; you will also have to tweak the systematic and unsystematic risk aversion parameters (RAP) to keep the same factor variances; you might also want to add a “pure stock-specific risk” asset to the risk model.

However, the more “hedged” the fund, the less it works.

Who should you listen to?
A sector decomposition of surprise stock returns
G. de Rossi (UBS) Northfield (2007)

A long-short strategy based on analysts’ earnings estimates upgrades and downgrades is profitable in some sectors but not in others, regardless of the quality of the forecasts (?). To explain this, one can start with a discounted cash flow model

$$\text{Price} = E \left[\sum_{t \geq 1} \frac{D_t}{1 + r_{0 \rightarrow t}} \right]$$

and try to separate the influence of the numerator (cash flow news) and the denominator (discount-rate news).

(I am not the best person to talk about discounted cash flow models: if you want more information, check the article itself.)

In a nutshell: in your models, use sector-specific weights.

Alpha scaling revisited Amish Shah
(Northfield, 2007)

The *Grinold formula* can be derived as follows: let

y :forward returns

g :investment signal (our alpha – or even its components)

\hat{g} :a realization of g

The ordinary least squares (OLS) estimate of y given $g = \hat{g}$ is

$$\begin{aligned} \hat{y} &= E[y] + \text{Cov}(y, g) \text{Cov}(g, g)^{-1} (\hat{g} - E[g]) \\ &= E[y] + \text{Cor}(y, g) \text{Sd}(y) \text{Sd}(g) \text{Sd}(g)^{-2} (\hat{g} - E[g]) \\ &= E[y] + \text{Cor}(y, g) * \text{Sd}(y) * (\hat{g} - E[g]) / \text{Sd}(g) \end{aligned}$$

Where

$\text{Cor}(y, g)$: information coefficient

$\text{Sd}(y)$: volatility

$\frac{\hat{g} - E[g]}{\text{Sd}(g)}$: score

This remains valid for time series regressions or cross-sectionnally (for the whole universe or within sectors).

The Northfield alpha scaling tool processes the client’s alpha as follows:

- gaussianize it to get a score
- use the cross-sectional Grinold formula:

$$\text{return forecast} = \text{IC} \times \text{volatility} \times \text{score}$$

The IC is provided by the user (perhaps on a sector basis); the volatility was not clearly explained (you could take the volatility from the risk model or compute a cross-sectional volatility for the whole market or for each sector).

This talk also mentioned the Black–Litterman framework. J. Sefton explained that this helps alleviate the “error maximization problem”: the risk factors used in the risk model are also present in the alpha and the optimizer is asked to minimize the former and maximize the latter – it ends up maximizing the misalignment between the two.

**Portfolio construction
in a regression framework
J. Sefton (formerly UBS)
Northfield (2007)**

Mean-variance optimization is often implemented as a 2-step process: first estimate the returns and the variance, then optimize. This can be replaced by a 1-step procedure, that goes directly from the data to the portfolio weights: constrained linear regression.

$$\begin{aligned}\gamma 1_T &= R w + \varepsilon \\ 1'_N w &= 1\end{aligned}$$

where

$$\begin{aligned}1_T &= (1 \cdots 1)' \\ 1_N &= (1 \cdot 1)' \\ R &= \begin{pmatrix} r_{11} & \cdots & r_{1N} \\ \vdots & & \vdots \\ r_{T1} & \cdots & r_{TN} \end{pmatrix}\end{aligned}$$

and γ is the risk appetite.

As a bonus, you get all the machinery surrounding regression: confidence intervals, robust regression, regression diagnostics (leverage, etc.) – the *audit trail* from data to weights is easier to follow.

For instance, looking at the *leverage* over time shows that, because of the recent drop in volatility, recent observations have a lower leverage, *i.e.*, a lower influence on the final weights, than older observations – this is not what you want.

Other example: the F-test can be used to test the difference between a portfolio P and an efficient portfolio E_γ ; it simplifies to

$$Z(\gamma) = \frac{\text{IR}(E_\gamma)^2 - \text{IR}(P)^2}{1 + \text{IR}(P)^2}$$

and the “distance” between portfolio P and the efficient frontier can be defined as

$$Z = \min_{\gamma} Z(\gamma).$$

I am not convinced that all the classical regression tools and formulas remain valid for constrained regression,

though. For confidence intervals, Dan di Bartolomeo wrote an article, 10 years ago.

The constrained regression can include priors for the returns and the risk matrix:

$$\begin{aligned}\gamma(1 \cdots 110 \cdots 0) &= \begin{pmatrix} R \\ \alpha_1 & \cdots & \alpha_N \\ V_{\text{prior}}^{1/2} \end{pmatrix} w + \varepsilon \\ \text{Var } \varepsilon &= \sigma^2 \text{diag}(1, t_0^{-1}, t_0^{-1})\end{aligned}$$

where t_0 is the “weight” (in days) of the prior.

**Distinguishing between being unlucky and
unskillful
Dan diBartolomeo (Northfield, 2007)**

Alpha or information ratios (IR) are rarely statistically significant, unless you have several centuries of data – and increasing the frequency of the data does not help: not only do the data become statistically more complicated (autocorrelation, kurtosis, jumps), but it does not increase the sample size, *i.e.*, the number of investment decisions taken. To account for regime switches, the size of the sample period can be estimated using the *CUSUM* method.

Grinold’s fundamental law of active management,

$$\begin{aligned}\text{IR} &= \text{IC} \sqrt{\text{breadth}} \\ \text{IR} &= \text{Information Ratio} = \frac{\alpha}{\text{tracking error}} \\ \text{IC} &= \text{Information coefficient} \\ &= \text{Cor}(\text{forecast, future returns}) \\ \text{breadth} &= \text{number of independent bets}\end{aligned}$$

makes unreasonable assumptions (no constraints, no transaction costs) and can be modified as

$$\text{IR} = \text{IC} \times \text{TC} \times \sqrt{\text{breadth}}$$

where the transfer coefficient $\text{TC} < 1$ measures portfolio construction efficiency.

Using the IR in the first place is not that good an idea: it only corresponds to a utility function if you are very, very risk averse – in which case, you should not approach stocks.

One can measure performance in a cross-sectional way (the sample size grows quickly) using the *effective information coefficient* (EIC, you can also see it as an *ex post transfer coefficient*): the correlation between the implied alpha (*i.e.*, the alpha for which the portfolio actually held would be optimal) and the realized returns.

Going the other way round, you can use the EIC to evaluate a risk model:

$$\text{excess returns} = \text{EIC} \times \text{dispersion} + \text{residuals}$$

where the dispersion is computed cross-sectionally and the dispersion of the residuals measures the quality of the risk model.

Good fund returns can have two causes: either a good *batting average* (the proportion of forecasts with the

correct sign) or a good *skew* (third moment of the product weights * returns), *i.e.*, larger gains than losses – the skew appears with trend-following strategies or portfolio insurance.

Someone remarked that those measures did not take *capacity* problems into account.

A market impact model that works
Dan diBartolomeo (Northfield, 2007)

Northfield now have a market impact model:

$$M = A + B * X + C * X^{1/2}.$$

Since large trades are extremely rare in the data (they are split into smaller trades), those model typically do not predict their impact well – the impact can exceed 100%. To avoid this, they add boundary constraints on the value of the coefficients, using a worst-case scenario (a takeover with information leakage). They also overweigh large trades (using their dollar value), in order to have better forecasts for them. Their current model works well for North America, but not for the UK or Japan – this may be due to a data problem.

In the Carisma seminar, Dan had more time and gave more details:

- When estimating the worst-case market impact, they take the liquidity into account
- Their market impact model has a liquidity exhaustion term

The market efficiency in stock markets
J.S. Yang et al.
arXiv:physics/0701179

Yet another econophysics article. For each asset of interest (here, indices: S&P 500 and Kospi) plot, over time (using high-frequency (1-minute) log-returns; only intraday to avoid discontinuities):

- The *tail index*;
- The “variance of the autocorrelation”, *i.e.*, the sum of the (day) 10 first autocorrelations, *i.e.*, the *Q-statistic* (used in the Ljung–Box test);
- The scaling property of the standard deviation, *i.e.*, μ in $\sigma(\Delta t) \sim (\Delta t)^\mu$.

Discretize the returns and compute:

- Their *entropy* $H(1) = \sum_i -p_i \ln p_i$;
- The entropy $H(L)$ of the sequences of L returns, or rather the marginal entropy $H(L) - H(L - 1)$;
- The *statistical complexity* (?) of the corresponding Markov chain (it will be zero if the data is either regular or completely random).

A valuation-based test of market-timing
W.B. Elliott et al.
Journal of corporate finance (2007)

When deciding whether to issue stock or debt, CFOs try to time the market. To test this hypothesis, one can

compute the correlation between the *leverage* (debt-to-equity ratio) and a misvaluation measure, such as the price-to-book ratio (P/B) – but the P/B contains more information than simple over-valuation: growth opportunities, asymmetric information, debt overhang, etc. Instead, the authors use the *residual income model* (RIM) to study the capital structure – CFOs do try to time the market.

CFA Institute magazine
March-April 2007

It does not make sense for a company not to pay dividends – it prevents investors from valuing it, unless you are satisfied with a null value.

Fat tails, power laws, drawdown can be used as a risk measure:

- APT models take fat tails into account;
- higher moments can be used in portfolio theory and lead to a *multi-moment efficient frontier*;
- The tails are different for market risk (thin) and credit and operational risk (fat); furthermore, they are not independent.

The article about the *quantitative investment process* (what we are doing) stressed the importance of sector-dependant, dynamic models; it also mentions *fundamental indices*, such as RAFI (Research Associates Fundamental Index), which weigh stock using (additive) variables other than market capitalization.

Fundamental indexation
R.D. Arnott et al.
Financial analysts journal (2005)

Extreme value problems in random matrix theory and other disordered systems
G. Biroli et al.
arXiv:cond-mat:0702244

There are several variants or generalizations of the central limit theorem:

- The asymptotic behaviour of $\sum_{i=1}^N x_i$ is gaussian if x is L^2 , stable (if defined) otherwise: only the tails matter;
- The asymptotic behaviour of $\text{Max}_{i=1}^N x_i$ is Weibull (if X is bounded), Gumbel (if X has thinner than power law tails) or Fréchet (if X has power law tails);
- The asymptotic behaviour of

$$S_q(N) = \sum_{i=1}^N x_i^q$$

- depends on $\mu = \sqrt{2 \ln N} / q\sigma$ and exhibit phase transition phenomena;
- etc.

The article examines what happens to the *Tracy–Widom distribution* (the distribution of the largest eigenvalue of a symmetric random matrix with gaussian entries) when the entries have power law tails.

On the distribution of the largest principal component
I.M. Johnstone (2000)

The largest singular of a random (gaussian) matrix (or the largest eigenvalue of a Wishart matrix) approaches the Tracy-Widom distribution, which is computable in terms of the Painlevé differential equation.

(Random matrix theory (RMT) is just a limit theorem for matrices, similar to other limit theorems (central limit, stable distributions, extreme distributions, large deviations, etc.)

Correlation functions, cluster functions and spacing distributions for random matrices
C.A. Tracy and H. Widom
arXiv:solv-int/9801004

Exponential weighting and matrix-based filtering of financial covariance matrices for portfolio optimization
S. Pafka et al.
arXiv:cond-mat/0402573

Random matrix theory (RMT) noise undressing can be combined with exponential weighting to account for heteroskedasticity.

Non-hermitian random matrices
B. Khoruzhenko (2001)

The tools used to study the (non-real) eigenvalues of non-hermitian matrices are very different.

Records in a changing world
J. Krug
arXiv:cond-mat/0702136

Records (*i.e.*, entries larger than all the previous ones) in a sequence of iid random variables asymptotically follow an extreme value theory (EVT) distribution and the distribution of record times do not depend on the distribution of the random variables.

One can also consider independant non-iid variables:

- $X_n = \text{Max}\{Y_{1,n}, \dots, Y_{N_n,n}\}$, with $Y_{i,j}$ iid, can be iused to mode sports records where N_n (usually increasing) is the population size;
- $X_n = Y_n + cn$;
- $X_n = \lambda_n Y_n$.

***Random but not so much
A parametrization for the returns and correlation matrix of financial time series***
A.C.R. Martins
arXiv:physics/0701025

Random matrix theory (RMT) can be seen as a statistical test of $H_0 : V = \mathbf{1}_{nn}$ using Spec \hat{V} , with no well-defined alternative hypothesis.

One can extend the Marčenko–Pastur model by:

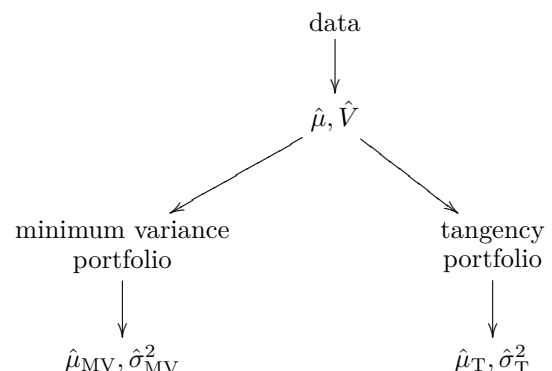
- allowing for a more general variance matrix V , e.g., one with a handful of non-zero eigenvalues (this is how RMT is used, to compare $H_0 : \lambda_k = 0$ against $H_1 : \lambda_k \neq 0$; but people use results valid for $k = 1$ for any value of k , and feel free to perform as many tests as possible);
- allowing for non-stationarity effects (the subject of the article, but their model is not very clear).

***Stock market distributions:
from past to present***
S. Drożdż et al.
arXiv:0704.0664

Earlier studies claimed that market fluctuations had power laws with a scaling index $\alpha > 3$, regardless of the horizon. This index now depends on the horizon: 3 for high frequency data (minute), while weekly data are gaussian.

Mean-variance portfolio analysis under parameter uncertainty
T. Bodnar and W. Schmid

Going from sample returns to market parameters to the (minimum variance or tangency) efficient portfolio to an estimation of the portfolio return and risk are estimators.



Under gaussian assumptions, one can compute, in closed form, the distribution of those estimators, and perform *tests* on them.

Note that

$$E[|\hat{\mu}_{MV}|] = \infty$$

$$E[|\hat{\sigma}_{MV}^2|^{1/2}] = \infty.$$

The minimum variance portfolio is more robust.

SNP: a program for non-parametric time series analysis
A.R. Gallant and G. Tauchen (1990–2007)

Hermite probability distribution functions (pdf) generalize the gaussian distribution:

$$h(z) \propto P(z)^2 \phi(z)$$

where P is a polynomial and ϕ the (multivariate, standard) gaussian pdf.

Semi-non-parametric distributions are the distributions of affine transformations of Hermite random variables.

***Conditional properties of hedge funds:
evidence from daily returns***

Y. Li and H. Kazemi

European financial management (2007)

Non-parametric GARCH(1,1) (or more generally SNP models) can be used to model daily hedge fund returns and accounts for asymmetries.

***Investor attention
and time-varying comovements***

L. Pend et al.

European financial management (2007)

Comovement can be measures as

$$\frac{\text{stock-specific risk}}{\text{total risk}}.$$

The authors use realized log-volatility from 5-minute returns and regress stock returns against S&P 500 returns.

A comparative study of portfolio insurance
S. Basak (2001)

Portfolio insurance can be cast into a utility framework: as the wealth approaches the floor, the marginal utility tends to infinity, e.g., one can use

$$\frac{(\text{wealth} - \text{floor})^\gamma}{\gamma}$$

instead of $\text{wealth}^\gamma/\gamma$.

The article uses continuous-time processes

***A shrinkage approach to model uncertainty
and asset allocation***
Z. Wang (2003)

To account for model uncertainty, shrink stock returns and variance towards those implied by the CAPM or Fama–French model.

Valuation in US commercial real estate
E. Ghysels et al.

European financial management (2007)

Real estate (and real estate investment trusts (REIT)) is very similar to equity: properties (especially commercial properties) can be valued with the *discounted rent model*, analogue of the discounted cash flow model; the *cap rate* (rent-to-price ratio) is the analogue of the P/E, etc.

***Is coskewness a better measure of risk in the
downside that downside beta?***
Evidence in emerging market data
D.U.A. Galagedera and R.D. Brooks
Journal of multinational financial management
(2007)

The various *downside betas*

$$\beta^{\text{HW}} = \frac{E[(r - r_{\text{free}})(r_{\text{market}} - r_{\text{free}})_{-}]}{E[(r_{\text{market}} - r_{\text{free}})_{-}^2]}$$

$$\beta^{\text{HR}} = \frac{E[(r - \bar{r})(r_{\text{market}} - \bar{r}_{\text{market}})_{-}]}{E[(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^2]}$$

$$\beta^{\text{E}} = \frac{E[(r - \bar{r})_{-}(r_{\text{market}} - \bar{r}_{\text{market}})_{-}]}{E[(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^2]}$$

can be generalized into *downside skewnesses*:

$$\gamma^{\text{HW}} = \frac{E[(r - r_{\text{free}})(r_{\text{market}} - r_{\text{free}})_{-}^2]}{E[(r_{\text{market}} - r_{\text{free}})_{-}^3]}$$

$$\gamma^{\text{HR}} = \frac{E[(r - \bar{r})(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^2]}{E[(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^3]}$$

$$\gamma^{\text{E}} = \frac{E[(r - \bar{r})_{-}(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^2]}{E[(r_{\text{market}} - \bar{r}_{\text{market}})_{-}^3]}$$

This is appropriate when the returns distribution is skewed.

Portfolio selection with higher moments
C.R. Harvey et al.

Traditional portfolio selection has two drawbacks:

- It assumes that the expected utility (the authors confuse utility and expected utility) only depends on the first two moments;
- It assumes that the market parameters are known, *i.e.*, that the *estimation risk* will remain negligible.

To tackle the first problem, one can:

- consider the expected utility as a function of the first three moments, leading to a 2-dimensional efficient frontier;
- replace the variance by an asymmetric measure of risk.

To tackle the second problem, one can:

- shrink the market parameters;
- use bayesian estimators of the market parameters (this is a non-linear shrinkage);
- use robust estimators of the market parameters;
- shrink the “optimal” portfolio (Black–Litterman);
- add constraints on the weights/

The authors use bayesian estimators from a multivariate *skew gaussian* (the sum of a truncated gaussian and a (smaller) gaussian, up to affine transformations) and a higher-dimensional efficient frontier to account for both skewness and estimation risk.

Their presentation of utility theory, with their undefined notion of *predictive utility* (something between utility and expected utility), is confusing.

As often, there is also some “resampled frontier bashing”.

Robust mean-variance portfolio selection **C. Perret-Gentil and M.P. Victoria-Feser (2003)**

“Robust portfolio construction” refers to two different notions:

- either a worst-case optimization;
- or a classical mean-variance optimization using robust estimates of the mean and variance matrices.

This article focuses on the latter and proves that it suffices to robustify the inputs of the mean-variance machinery to get a robust mean-variance portfolio.

However,

- it does not work complicated (non-convex) constraints – there are similar limitations to the equivalence between robust optimization (first sense) and bayesian optimization;
- in high dimensions (the article remains in dimension 3), robust variance matrices are not very robust.

Hedging predictions in machine learning **A. Gammernan and V. Vovk (2007)**

While statistical prediction procedures can provide confidence intervals (or, even better, full posterior distributions, for bayesian statistics), machine learning algorithms only provide a single, naked prediction.

- Find a measure of strangeness, using a machine learning (ML) algorithm, in an ad hoc way, e.g.:
 - absolute value of the residuals in a regression;
 - Lagrange multipliers in a support vector machine (SVM) (it is zero for non-support vectors);
 - for the k nearest neighbours,

$$\alpha = \frac{\sum_{j=1}^k d_{ij}^+}{\sum_{j=1}^k d_{ij}^-}$$

where d_{ij}^+ (resp. d_{ij}^-) is the j th shortest distance (euclidian or other) between i and the observations with the same (resp. a different) label (this can be seen as a “truncated anova”);

- The *empirical p-value* $p_{x,y}$ of an observation (x, y) is the proportion of observations stranger than it (this should remind you of bootstrap p -values);
- For regression problems, after choosing a confidence level ε , the prediction of x is the set

$$\{y \in Y : p_{x,y} > \varepsilon\};$$

- For classification problems,

$$\hat{y} = \underset{y}{\operatorname{Argmax}} p_{x,y}$$

$$\text{credibility}(\hat{y}) = p_{x,\hat{y}}$$

$$\text{confidence}(\hat{y}) = 1 - \underset{y \neq \hat{y}}{\operatorname{Max}} p_{x,y}$$

(this should remind you en *entropy-based estimators*: we are looking for the model under which the data looks as random as possible, *i.e.*, the model that removes as much regularity as possible from the data); the credibility should be high, unless the training set is not iid and/or not representative of the data and/or the new observation is an outlier.

More formally, the measure of strangeness α is called a *non-conformity measure* and is just a family of maps $(X \times Y)^n / \mathfrak{S}_n \rightarrow \mathbf{R}_+$.

The empirical p -value is an approximate *random test*: ideally, it should be a function $t : \mathbf{Z}^* \rightarrow [0, 1]$, where \mathbf{Z}^* is the set of all finite sequences of integers, such that for all $\varepsilon > 0$, for all $n \in \mathbf{N}^\times$, for all probability distribution P on \mathbf{Z}^n ,

$$P(\{z \in \mathbf{Z}^n : t(z) \leq \varepsilon\}) \leq \varepsilon.$$

The article mentions *ridge regression* but seems unaware of the fact that this is not a single estimator but a family of estimators (a *regularization path*), depending on the value of the ridge parameter.

The style consistency of hedge funds **R. Gibson and S. Gyger** **European financial management (2007)**

PAM (partitionning around medoids – this is similar to k -means clustering but the center of the clusters are chosen among the observations) can recognize hedge fund styles as well as principal component analysis (same R^2).

Fuzzy clustering algorithms find the clusters and assign a membership probability for each observation-cluster pair. The authors use an *entropy-based* fuzzy clustering:

- Entropy-based estimators find the probability distribution (p_1, \dots, p_n) that maximizes the entropy $H = -\sum p_i \ln p_i$, subject to all the information available expressed as constraints.
- The cost of a configuration (*i.e.*, a set of membership probabilities) is

$$\text{cost}(Y) = \sum_{ij} p_{ij} d(r_i, y_j)$$

where the r_i are the observations and the y_j the cluster centers. If the cost is known, we can find the membership probabilities that maximize the entropy under the constraint above.

- For the cost defined by each hard clustering configuration (yes, there are many of them), compute the entropy-maximizing membership probabilities and

average them all (*i.e.*, you have no prior preference on the hard clustering configurations).

- Finally, with those membership probabilities, find the cluster centers by maximum likelihood.

After fuzzy clustering, style consistency of hedge funds can be measured as *mean time consistency* (average (over time) membership probability), *consistency deviation* (standard deviation of time consistency) or their ratio (*consistency ratio*); style consistency does not affect returns.

Calibration risk for exotic options
K. Detlefsen and W.K. Härdle
Journal of derivatives (2007)

Derivative models have to be calibrated by minimizing some error function, but the calibration depends on the choice of this error function, leading to *calibration risk* – on top of *model risk*.

Popular error functions include:

- absolute price difference;
- relative price difference;
- absolute implied volatility difference;
- relative implied volatility difference.

This article compares them for the *Heston model*

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t} dW_t^1$$

$$dV_t = \xi(\eta - V_t)dt + \theta\sqrt{V_t}dW_t^2$$

(well-defined if $\xi\eta > \theta^2/2$, but in simulations, negative variances are truncated to zero) and the *Bates model*, which accounts for jumps in the stock prices,

$$\frac{dS_t}{S_t} = \mu dt + \sqrt{V_t}dW_t^1 + dZ_t$$

$$dV_t = \xi(\eta - V_t)dt + \theta\sqrt{V_t}dW_t^2$$

$$Z \sim \text{Poisson}(\lambda, k)$$

$$\log(1+k) \sim N\left(\log(1+\bar{k}) - \frac{\delta^2}{2}, \delta^2\right).$$

The models are calibrated with vanilla options (using the characteristic function of those models – yet another application of the characteristic function) and then used to price exotic options: barrier (un and out call)

$$\text{Payoff} = \mathbf{1}_{\text{Max } S < B}(S_T - K)^+$$

and cliquet

$$\text{Payoff} = \sum_{i=1}^N \frac{S_{t_i} - S_{t_{i-1}}}{S_{t_{i-1}}}$$

(the cliquet pays the sum of the *linear* returns; it is actually more complicated: in each period, the returns are truncated (above and below), and so is their sum).

The article suggests to calibrate with respect to absolute prices to mitigate model risk – or relative implied volatilities, if the model has already been chosen.

Hedge fund activism, corporate governance
and firm performance
A. Brav et al.

Some hedge funds can acquire more than 5% of a company (this has to be filed to the SEC (schedule 13D)) and ask it to change (e.g., sell part of its business, fire its CEO, accept a takeover). The following variables can help predict targeted companies:

- size:
 - market value;
- value:
 - $Q = \frac{\text{debt} + \text{market value}}{\text{debt} + \text{book value}}$;
 - book-to-market;
- performance:
 - sales growth;
 - $\text{ROA} = \frac{\text{EBITDA}}{\text{assets}}$;
- capital structure:
 - debt-to-capital;
 - new equity issuance;
 - dividend yield;
 - payout ratio = $\frac{\text{dividends}}{\text{net income before extraordinary items}}$;
- governance:
 - number of Ginex takeover defenses;
 - institutional ownership;
- liquidity:
 - Amihud illiquidity (daily values, averaged over 1 year):

$$E \left[0.001 \cdot \sqrt{\frac{\text{price} \cdot \text{volume}}{|\text{returns}|}} \right]$$

- idiosyncratic volatility: monthly standard deviation of the daily returns in excess of the industry mean.

A breakdown of the valuation effects of
international cross-listing

A. Bris et al.
European financial management (2007)

Cross-listing effects can be explained by:

- market segmentation;
- liquidity;
- bonding (differences in rights);
- signaling.

Acquisitions, overconfident managers
and self-attribution bias

J.A. Doukas and D. Petmezas
European financial management (2007)

Managers with more than 5 acquisitions in the past 3 years are overconfident.

***Do life insurance stocks
provide superior returns?***
M. Najand et al.

Journal of asset management (2007)

CAPM-GARCH (*i.e.*, CAPM with GARCH noise) is better than CAPM.

Managers with more than 5 acquisitions in the past 3 years are overconfident.

Risk and asset allocation
A. Meucci (2006)

Chapter 1: Univariate statistics

There are three equivalent ways of describing a univariate distribution: the probability distribution function (pdf, f_X), the cumulative distribution function (cdf, F_X) and the *characteristic function* ($\phi_X(\omega) = Ee^{i\omega X}$) – there is also the *quantile function* (F_X^{-1}), but it does not usefully generalize to higher dimensions.

A *location parameter* (mean, median, mode) is defined by affine equivariance; so are *dispersion parameters*: for instance, the standard deviation, the MAD, the interquartile range or the *modal dispersion*

$$\text{MDis} = \left(\frac{d^2 \ln f_X}{dx^2} \Big|_{x=\text{Mod } X} \right)^{-1}.$$

The *Z-score* for a location and a dispersion parameters is

$$Z = \frac{X - \text{Loc } X}{\text{Dis } X}.$$

The first chapter ends with a taxonomy of univariate distributions: uniform, gaussian, Cauchy, Student, log-normal, Gamma (a generalization of the χ^2 distribution, often used as a prior for variance).

Chapter 2: Multivariate statistics

In a multivariate setting, there are still three ways of representing a distribution: pdf, cdf and characteristic function. A multivariate distribution can be factored into 1-dimension distributions (the marginals) and a “purely joint component” – the *copula*, defined as the joint definition of its grades (the *grade* of a univariate distribution being its uniformization).

The book fails to give a taxonomy of copulas, but provides a few examples:

- The copula of a log-normal distribution is a gaussian copula;
- The Student T copula is not independant;
- The copula between prices and log-prices, or between ratio-returns and log-returns, is trivial;
- The copula between a call option and its underlying is trivial.

The copula is invariant under monotonic increasing transformations – e.g., replacing a stock by a call option has no effect.

A *location parameter* (mean, mode) is characterized by

affine equivariance:

$$\text{Loc}(a + BX) = a + B \text{Loc } X$$

for all invertible affine transformations $x \mapsto a + Bx$.

In dimension greater than 1, the median is not a location parameter: it depends on a choice of coordinates.

A *dispersion matrix* is a symetric, positive matrix satisfying the affine equivariance property:

$$\text{DisSq}(a + Bx) = B \text{DisSq}(X) B'$$

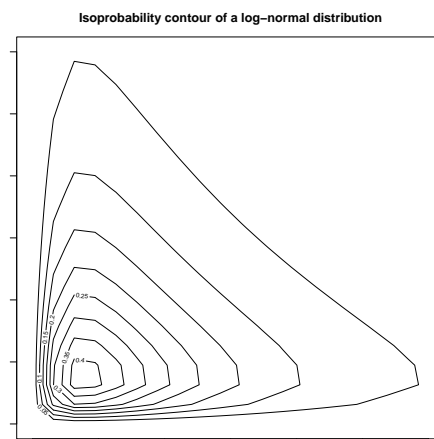
for all invertible affine transformations $x \mapsto a + Bx$, such as the covariance matrix or the *modal dispersion*:

$$\text{MDis } X = - \left(\frac{\partial^2 \ln f_X}{\partial x \partial x'} \Big|_{x=\text{Mod } X} \right)^{-1}.$$

The *Z-score* is then:

$$Z = \sqrt{(X - \text{Loc } X)' (\text{DisSq } X)^{-1} (X - \text{Loc } X)}.$$

Any measures of location and dispersion define a family of *location-dispersion ellipsoids* – for non-elliptical distribution, this information is far from sufficient.



For a more empirical/non-parametric/procedural analogue of the location-dispersion ellipsoid, check the notions of *bag plot* and *half-space depth*.

The expected value and the variance are equivariant with respect to not-necessarily invertible transformations: you can obtain the location-dispersion ellipsoid of a portfolio from that of the market.

Higher-order statistics (HOS) can also be defined in the multidimensional case, but they are *tensors*; however, the *coskewness* and the *cokurtosis* tensors can be summarized in one overall index of symmetry (for coskewness) or tail thickness (for kurtosis) – but the book does not tell us how...

The *Schweizer–Wolff* measure of *dependence* of two (univariate) random variables is the L^p distance between their copula and the independant copula, normalized to take values in $[0, 1]$; it is invariant under monotonic (increasing or decreasing) transformations.

Measures of *concordance*, such as *Kendall's* τ or *Spearman's* ρ are only invariant under increasing transformations and take values in $[-1, 1]$; they change sign

under decreasing transformations; furthermore, being zero is only a necessary condition for independence.

Correlation mixes marginal and joint features: it is only an “affine measure of concordance”, it is changed by monotonic transformations of the marginals: “One might wonder why correlation is such a popular tool: (...) for an important class of distributions [elliptical distributions], the correlation completely defines the dependance structure.”

The taxonomy of multivariate distributions includes: matrix-variate gaussian and T distributions, Wishart (generalization of the χ^2 : it is the distribution of $X_1 X_1' + \dots + X_\nu X_\nu'$, where $X_i \sim N(0, \Sigma)$ with $\nu \geq n$).

Elliptical distributions are affine transformations of spherically symmetric distributions, which can be described by a single univariate function: $X \sim \text{El}(\mu, \Sigma, g)$ has density

$$f(x) = |\Sigma|^{-1/2} g((x - \mu)' \Sigma^{-1} (x - \mu))$$

where g is a *probability density generator*: $g \geq 0$ and $\int_0^\infty v^{N/2-1} g(v) dv < \infty$. For instance:

$$g(z) \propto e^{-z/2} \quad (\text{gaussian})$$

$$g(z) \propto (1+z)^{-(1+N)/2} \quad (\text{Cauchy})$$

$$g(z) \propto \left(1 + \frac{z}{\nu}\right)^{-(\nu+N)/2} \quad (\text{Student})$$

(Notice that the gaussian pdf has an exponential decay, while the Student or Cauchy pdf have a *power law* decay.)

Alternatively, elliptic distributions can be recognized from their characteristic function, $\phi(\omega) = e^{i\omega' \mu} \psi(\omega' \Sigma \omega)$, for some suitable real-valued function ψ .

When a *stable distribution* describes a phenomenon (log-returns, risk, etc.) at horizon T , the compounded distribution at horizon $2T$, $3T$, etc. is still in the same family. *Symmetric-alpha-stable* (sas) distributions, $X \sim \text{SS}(\alpha, \mu, m_\Sigma)$, are defined as

$$\phi_X(\omega) = e^{i\omega' \mu} \exp \left(- \int_{\mathbf{R}^N} |\omega' s|^\alpha m_\Sigma(s) ds \right)$$

where m_Σ is a symmetric measure on the ellipsoid $s' \Sigma^{-1} s = 1$.

Beware, stable distributions are dangerous: they violate the central limit theorem (they have no second moment).

When an *infinitely divisible* distribution describes some phenomenon at horizon T , you can (under an independence assumption) get the distribution at horizons $0 < t < T$.

Chapter 3: Market invariants

To model a market, the author suggests to:

- Look for *market invariants*, *i.e.*, iid random variables built from market data (if you see the market

as a machine to produce prices from iid noise, the noise is a market invariant – some people speak of *innovations*), recognized by looking (graphically) at their autocorrelation and comparing their distribution in the first and last half of the sample; examples include: log-returns for the equity market, changes in yield to maturity for the fixed income market, changes in (ATMF) implied volatility for the options market;

- Model their distribution;
- Project their distribution to the investment horizon (e.g., we could use daily data to invest on a monthly horizon) – this can be done using the *characteristic function* and is even easier with additive invariants (such as log-returns);
- Transform the projected market invariant distribution into a market price distribution – even if that transformation is not analytically tractable, one can easily get the moments of the distribution of prices – but be sure to keep track of the propagation of estimation errors.

The quest for market invariants can involve *dimension reduction* or *variable selection*, but this part of the book is a bit confusing and the author fails to warn the reader of the dangers of model selection.

Cointegration is mentioned, for the equity and fixed income markets, but not detailed.

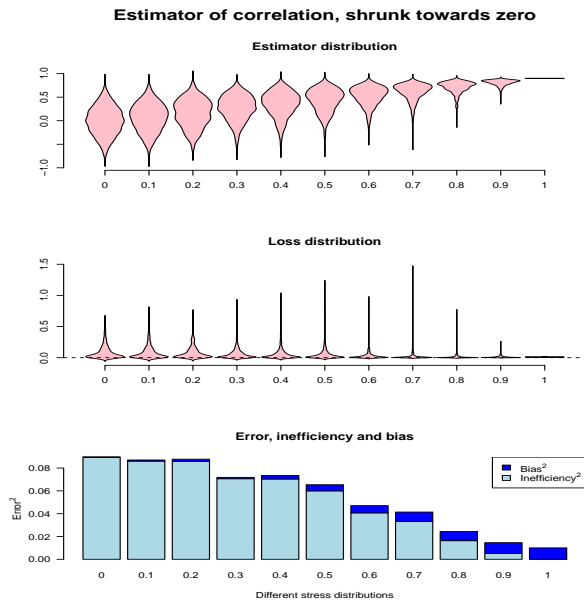
Chapter 4: Estimators

Depending on the amount of information available, you will prefer shrinkage or bayesian estimates (very little data), maximum likelihood estimators (MLE) (more data) or non-parametric estimators (a lot of data).

The quality of an estimator can be measured as its *error* (mean square error, MSE), its *bias* (average distance to the correct value) and its *inefficiency* (standard deviation, *i.e.*, dispersion around its expected value):

$$\text{Error}^2 = \text{Bias}^2 + \text{Inefficiency}^2.$$

One can also consider the *loss*, *i.e.*, the squared distance to the correct value (with respect to some quadratic form) – contrary to the error, the bias or the inefficiency, this is not a number but a random variable. One can look at these quantities and distributions for a family of “stress-test distributions”, to gauge estimation risk.



The *Glivenko–Cantelli* theorem states that the empirical cumulative distribution function (ecdf) is a consistent estimator of the cdf; this yields the *benchmark (non-parametric) estimator*: just replace the cdf in the definition of the quantity of interest by the ecdf; those estimators are often a good starting point and can be improved on (e.g.: sample mean and OLS benchmark estimators are non-biased, while the variance benchmark estimator is).

Kernel estimators are obtained from the benchmark estimators by replacing the Dirac masses in the epdf by gaussian kernels – *i.e.*, we use a smoothed estimator of the pdf.

MLE relies on the assumption that the distribution from which the data is drawn is in a very small set of (known) distributions (sometimes called the *stress distributions*): the MLE is the mode of the distribution of the parameters given the data; it is asymptotically unbiased, asymptotically gaussian, and asymptotically attains the Cramer–Rao bound (*i.e.*, it is the best unbiased estimator – the book assumes you are already familiar with those notions).

The MLE of the parameters of an elliptic distribution (with a known probability density generator g) is a weighted mean and weighted variance; the weights can be computed (iteratively) using the Mahalanobis distance and the density generator. The book does not consider the (semi-parametric) situation of an elliptic distribution with an unknown probability density generator. Linear regression or principal components are still amenable in the case of elliptic distributions.

In the gaussian case, the *condition number*, *i.e.*, the ratio of the smallest to the largest eigen value, measures how close to a sphere the cloud of points is: if it is close to 1, the problem is *well-conditionned*, if it is close to 0, it is *ill-conditionned*.

An estimator is *admissible* if there does not exist another estimator with a lower error for all the stress-test distributions. Benchmark estimators or MLE estima-

tors tend not to be admissible: their bias is low but their inefficiency large (especially when the condition number is close to one); shrinkage estimators have a larger bias and often a much smaller inefficiency, resulting in a lower error.

For instance, the sample mean, with gaussian data, in dimension at least 2, is not admissible: the *James–Stein shrinkage estimator*

$$\hat{\mu}^{\text{Stein}} = (1 - \alpha)\hat{\mu} + \alpha\mathbf{b}$$

$$\alpha = \frac{1}{T} \frac{N\bar{\lambda} - 2\lambda_1}{(\hat{\mu} - \mathbf{b})'(\hat{\mu} - \mathbf{b})}$$

where T is the number of observations, N the dimension, λ_1 the largest eigenvalue (we do not know it: it will have to be computed from an estimator of the variance matrix), $\bar{\lambda}$ the average eigen value and \mathbf{b} any vector, is admissible. The prior \mathbf{b} can be chosen arbitrarily (e.g., 0), or using “prior” information, or as the *grand mean*

$$\mathbf{b} = \frac{\mathbf{1}'\hat{\mu}}{N}\mathbf{1}$$

or as the volatility-weighted grand-mean (*Jorion estimator*)

$$\mathbf{b} = \frac{\mathbf{1}'\hat{\Sigma}^{-1}\hat{\mu}}{\mathbf{1}'\hat{\Sigma}^{-1}\mathbf{1}}\mathbf{1}.$$

(Strictly speaking, this is not prior information, since it is extracted from the same data set; but it is much less volatile.)

Dispersion estimators can be assessed with the *Frobenius loss*

$$\text{Loss}(\hat{\sigma}, \sigma) = \text{tr} \left(\hat{\Sigma} - \Sigma \right)^2.$$

The sample covariance matrix scatters the eigen values away from $\bar{\lambda}$ (this is easily seen when the true eigen values are all equal: the sample eigen values will be more dispersed, the largest eigen values will be too large and the smallest eigen values too small): it squeezes and stretches the location-dispersion ellipsoid; the estimation worsens the condition number of the market invariants. One can shrink the eigenvalues towards their mean (Ledoit):

$$\hat{\Sigma}^s = (1 - \alpha)\hat{\Sigma} + \alpha\hat{C}$$

$$\hat{C} = \frac{1}{N} \sum_{n=1}^N \hat{\lambda}_n$$

$$\alpha = \text{Min} \left(1, \frac{\frac{1}{T} \sum_t \text{tr}(x_t x_t' - \hat{\Sigma})^2}{\text{tr}(\hat{\Sigma} - \hat{C})^2} \right)$$

Ledoit shrinkage also works for regression.

Robustness

Here are a few measures of robustness:

- the *leave-out-one Jackknife*;
- the *sensitivity curve* (add an observation instead of removing it; this is a function of the added observation; in particular, we are interested whether this function is bounded);

- the *influence function* (the infinite-sample limit of the sensitivity curve): generalize your estimator so that it be a function of a distribution:

$$\hat{G} = G\left[\sum_t \delta^{(x_t)}\right]$$

and consider its *Gâteaux derivative* (or *directional derivative*: contrary to the usual (Fréchet) derivative, we do not require it to be linear, *i.e.*, the derivative in the direction $\vec{u} + \vec{v}$ need not be the sum of the derivatives in the directions \vec{u} and \vec{v}):

$$\text{IF}(x, f, \hat{G}) = \lim_{\varepsilon \rightarrow 0} \frac{1}{\varepsilon} \left(\hat{G}[(1 - \varepsilon)f + \varepsilon \delta^{(x)}] - \hat{G}[f] \right).$$

For maximum likelihood estimators (MLE), one can show that

$$\text{IF}(x, f, \hat{\theta}) = A \left. \frac{\partial \ln f_{\theta}}{\partial \theta} \right|_{\theta=\hat{\theta}[f]}$$

$$A = - \left[\int \left. \frac{\partial^2 \ln f_{\theta}(x)}{\partial \theta \partial \theta'} \right|_{\theta=\hat{\theta}[f]} f(x) dx \right]^{-1}.$$

The influence functions of the gaussian MLE of location and dispersion are not bounded: they are not robust; for other elliptical distribution, they are.

More generally, *M-estimators* (or *generalized MLE*) are obtained by modifying the log-likelihood so that the influence be bounded; M-estimators of location and dispersion are actually weighted means and variances.

Outliers

The *breakdown point* of an estimator is the proportion of outliers it can sustain while maintaining a bounded influence function.

The *minimum volume ellipsoid* (MVE: find the smallest ellipsoid containing $x\%$ of the data, for various values of x ; if there is a jump in the volume, we know there are outliers and we know how many) and the *minimum covariance determinan* (MCD: the minimum value of the determinant of the sample covariance matrix containing $x\%$ of the data) have a high breakdown point and are usually computed in a greedy and approximate way, by discarding the observations one at a time.

Missing data

To compute location and dispersion parameters in presence of *missing values*, one can do better than discarding incomplete observations. In dimension 2, if the time series are complete but have different lengths, there is an explicit formula (Stambaugh). In the general case, use the *expectation-maximization* (EM) algorithm.

Beware: the EM algorithm is not a data imputation algorithm. Indeed, since the values are replaced by their expected values, they will have a much lower dispersion; in particular, you cannot naively use them to compute a dispersion parameter (the EM algorithm explains how to compensate for that in the case of the variance – some books forget that compensation in their presentation of the algorithm).

Weighted estimates

Since more recent observations contain less stale information, they can be given a linear weight, either with a *moving window* or *exponential smoothing* – to find the *decay factor*, just put it in the log-likelihood formula for $\hat{\mu}$ and $\hat{\Sigma}$ – this is actually consistent with a GARCH model.

If the location is known to be close to zero (with respect to the dispersion), you can assume it actually is zero: this is a *shrinkage estimator*.

One can also estimate location, dispersion or any other parameter using a *pricing model*, *i.e.*, choosing the parameters so that the model-implied price be as close as possible from the market price.

Chapter 5: Evaluation allocations

The *investor's objective* can be expressed in monetary terms: usually the final wealth or the change in wealth (when the absolute value of wealth does not matter) or the difference between the final wealth and a benchmark.

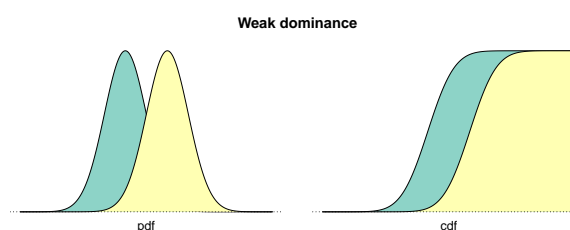
Dominance

An allocation α *strongly dominates* an allocation β if $\Psi_{\alpha} \geq \Psi_{\beta}$ a.s, where Ψ_{α} is the investor's objective coming from allocation α (it is a random variable). Strong dominance (also called *zeroth-order dominance*) rarely happens and relies on the *join* distribution $(\Psi_{\alpha}, \Psi_{\beta})$.

Weak dominance (or *first-order dominance*) is defined as

$$\forall \psi \in \mathbf{R} \quad F_{\Psi_{\alpha}}(\psi) \leq F_{\Psi_{\beta}}(\psi)$$

where F is the cumulative distribution function.



Equivalently:

$$\forall p \in [0, 1] \quad Q_{\Psi_{\alpha}}(\psi) \geq Q_{\Psi_{\beta}}(\psi)$$

where Q is the quantile function.

Second-order stochastic dominance (SSD) is defined as

$$\forall \psi \in \mathbf{R} \quad E[\Psi_{\alpha} - \psi]^{-} \geq E[\Psi_{\beta} - \psi]^{-},$$

i.e., for every benchmark ψ , the underperformance of α is not as bad as that of β . Equivalently:

$$\forall \psi \in \mathbf{R} \quad I^2[f_{\Psi_{\alpha}}](\psi) \geq I^2[f_{\Psi_{\beta}}](\psi)$$

where

$$I^2[f_{\Psi}](\psi) = I[F_{\Psi}](\psi) = \int_{-\infty}^{\psi} F_{\Psi}(s) ds.$$

These definitions can be generalized to *order-q dominance* – but these are not easily interpretable and still fail to yield a total order.

Satisfaction

Instead of trying to assess the whole distribution of the investor's objective Ψ_α , one can try to summarize an allocation α into a single number $S(\alpha)$, an *index of satisfaction*. We might want it to satisfy some of the following properties:

- money-equivalence: it should be measured in units of money;
- sensibility, *i.e.*, consistence with strong dominance;
- consistence with stochastic dominance;
- constancy: if the distribution of the investor's objective is concentrated in a single point, $\Psi_\alpha = \delta_\psi$, then $S(\alpha) = \psi$;
- positive homogeneity of degree 1 (the investor's objective already is):

$$\forall \alpha \quad \forall \lambda > 0 \quad S(\lambda \alpha) = \lambda S(\alpha)$$

in particular, this gives the *contribution* of each asset (Euler's formula):

$$S(\alpha) = \sum_{n=1}^N \alpha_n \frac{\partial S(\alpha)}{\partial \alpha_n}$$

- translation invariance (translation by a deterministic allocation): if $\Psi_b = \delta_{\psi_b}$, then $S(\alpha + b) = S(\alpha) + \psi_b$;
- super-additivity (if you focus on risk and not satisfaction, you will call this *subadditivity*):

$$\forall \alpha, \beta \quad S(\alpha + \beta) \geq S(\alpha) + S(\beta)$$

- comonotonic additivity (two allocations α and β are *comonotonic* if their objectives are increasing functions of one another, for instance a stock and a call option on it): if α and β are comonotonic, then $S(\alpha + \beta) = S(\alpha) + S(\beta)$, *i.e.*, the index of satisfaction is option-proof;
- concavity (results from homogeneity and superadditivity), *i.e.*, the index of satisfaction promotes diversification;
- risk aversion: if $\Psi_b = \delta_\psi$ and $E\Psi_\alpha = 0$ then $S(b) \geq S(b + \alpha)$.

Examples include:

- the expected value of the objective, $S(\alpha) = E\Psi_\alpha$;
- the *Sharpe ratio*

$$SR(\alpha) = \frac{E\Psi_\alpha}{Sd\Psi_\alpha}$$

(it is not expressed in monetary units; it is homogeneous of degree 0, not 1);

- the certainty-equivalent;
- the value at risk;
- expected shortfall.

Certainty-equivalent

Given a (continuous) increasing *utility function* u , the *certainty equivalent* is

$$CE(\alpha) = u^{-1}E[u(\Psi_\alpha)];$$

this is the value of the investor's objective of a deterministic investment having the same expected utility as α – *i.e.*, this is the expected utility expressed in monetary units thanks to the (inverse of the) utility function:

- it is only homogeneous for *power utility*, $u(\psi) = \psi^{1-1/\gamma}$, $\gamma \geq 1$;
- it is not comonotonic additive (except for a linear utility);
- it is not superadditive (except for a linear utility);
- in general, it is neither nor convex (in particular, the convexity of the utility function is not linked to the convexity of the certainty equivalent);
- if the utility is concave, it is risk-averse.

For theoretical purposes, the utility function can be written

$$u(\psi) = \int g(\theta) H^{(\theta)}(\psi) d\theta, \quad g \geq 0, \quad \int g = 1$$

where H is the Heaviside function (this generates all increasing functions) or

$$u(\psi) = \int g(\theta) \text{Min}(\psi, \theta) d\theta, \quad g \geq 0, \quad \int g = 1$$

(which generates all concave functions).

(I do not understand the explanations about the investor's *subjective probability*.)

A utility function is entirely determined (up to a positive affine transformation) by its *Arrow-Pratt absolute risk aversion*,

$$A(\psi) = -\frac{u''(\psi)}{u'(\psi)}.$$

The *Pearson specification* includes the following particular cases:

- hyperbolic absolute risk aversion (HARA, always concave):

$$A(\psi) = \frac{1}{\gamma\psi + \zeta}$$

- exponential utility: $u(\psi) = -\exp -\frac{\psi}{\zeta}$
- quadratic utility (beware, it is not sensible for $\psi > \zeta$):

$$u(\psi) = \psi - \frac{\psi^2}{2\zeta}$$

- power utility: $u(\psi) = \psi^{1-1/\gamma}$
- logarithmic utility: $u(\psi) = \ln \psi$
- linear utility: $u(\psi) = \psi$.

The Arrow-Pratt risk aversion also yields approximations of the certainty equivalent and the *risk premium*:

$$CE(\alpha) \approx E\Psi_\alpha - \frac{1}{2}A(E\Psi_\alpha) \text{Var} \Psi_\alpha$$

$$\begin{aligned} RP(\alpha) &= E[\Psi_\alpha] - S(\alpha) \\ &= E[\Psi_\alpha] - CE(\alpha) \\ &\approx \frac{1}{2}A(E\Psi_\alpha) \text{Var} \Psi_\alpha \end{aligned}$$

In *prospect theory*, utility functions are S-shaped: concave (risk-averse) for profits and convex (risk-prone) for losses.

Quantile (VaR)

The *value at risk* (VaR), defined as

$$Q_c(\alpha) = Q_{\Psi_\alpha}(1 - c),$$

is

- not consistent with second-order dominance (the definitions of second-order dominance and of the expected shortfall are very similar);
- consistent with first-order dominance;
- not super-additive: it fails to promote diversity;
- comonotonic additive (it is not fooled by derivatives);
- positive homogeneous;
- neither concave nor convex;
- not risk-averse.

The VaR can be computed with: a lot of data, or a gaussian assumption, or the *Cornish-Fisher expansion* (which is an approximation of the quantile function, but it need not be particularly accurate for the extreme values we are interested in), or *extreme value theory* (EVT) (which requires enough data in the tails – but in the 1% tail, you only have 1% of your data).

Coherent indices, spectral indices, expected shortfall

An index of satisfaction is *coherent* if it is

- consistent with strong dominance;
- positive homogeneous;
- translation invariant;
- superadditive.

This implies money-equivalence and concavity. The *one-sided moments* are coherent (but they are not comonotonic additive).

A coherent index of satisfaction is *spectral* if it is estimable (?) and comonotonic additive. This implies weak stochastic dominance and risk aversion.

Spectral indices are of the form

$$\text{Spc}_\phi(\alpha) = \int_0^1 \phi(p) Q_{\Psi_\alpha}(p) dp$$

where ϕ , the *spectrum*, is decreasing, $\phi(1) = 0$ and $\int_0^1 \phi = 1$.

Chapter 6: Optimizing allocations

Portfolio construction proceeds as follows:

- Define the investor's objective;
- Define his index of satisfaction;
- Model the market invariants;
- Project the market invariants to the investment horizon;
- Gather other information: legal constraints, transaction costs;
- Maximize the investor's satisfaction – if there are

several indices of satisfaction, maximize the first subject to bounds on the others.

The author details an example with a closed form solution: total wealth, certainty equivalent for the exponential utility, gaussian prices (the certainty equivalent is then quadratic), linear transaction costs, constraint on the value at risk.

The classes of optimization problems that can be efficiently solved include, from the most specific to the most general:

- Linear programming (LP);
- Quadratic programming (QP);
- Quadratically-constrained linear programming (QCLP);
- Second-order cone programming (SOCP: the *ice cream* constraints);
- Semi-definite programming (SDP);
- Cone programming (up to here, interior point methods are available);
- Convex programming.

Most of the time, e.g., with value at risk or certainty equivalent, we are not that lucky: the optimization problem is not convex. The *mean-variance approximation* can make this problem amenable:

- Express the satisfaction as a function of the moments of the market distribution,

$$S(\alpha) = H(E[\Psi_\alpha], \text{CM}_2(\Psi_\alpha), \text{CM}_3(\Psi_\alpha), \dots)$$

(this is obtained from a Taylor expansion of the utility function or the Cornish–Fisher expansion of the value at risk); the problem is now infinite-dimensional:

$$\begin{array}{ccc} \mathbf{R}^n & \xrightarrow{\quad} & \mathbf{R}^N \xrightarrow{\quad} \mathbf{R} \\ \alpha \mapsto & (E[\Psi_\alpha], \text{CM}_2(\Psi_\alpha), \dots) \mapsto & S(\alpha) \end{array}$$

- Assume that the index of satisfaction is well approximated by the first two moments

$$S(\alpha) \approx \tilde{H}(E[\Psi_\alpha], \text{CM}_2(\Psi_\alpha))$$

- Since the index of satisfaction is consistent with weak stochastic dominance, the optimal allocation is on the *efficient frontier*

$$\alpha(v) = \underset{\substack{\alpha \in \mathcal{C} \\ \text{Var } \Psi_\alpha = v}}{\text{Argmax}} E[\Psi_\alpha], v \geq 0$$

- You now just have to maximize the index of satisfaction on the efficient frontier.

The approximation of $S(\alpha)$ from the first two moments is valid in the following cases:

- The market prices Ψ_α are elliptical (e.g., gaussian), so that their distribution is entirely determined by the first two moments (regardless of the index of satisfaction); this is wrong for derivatives and even for stock prices (it has to be prices, not their logarithms) – but for short horizons, this is good enough an approximation;

- The index of satisfaction really depends only on the first two moments (regardless of the distribution of the market invariants).

Contrary to what many people believe, the risk aversion parameter λ in $E[\Psi_\alpha] - \lambda \text{Var} \Psi_\alpha$ is not a feature of the investor and it does not define an index of satisfaction: its value also depends on the market – in some extreme cases, it only depends on the market and not on the investor; choosing it beforehand yields allocations inconsistent with strong dominance.

The mean-variance approximation is a two-step process: first compute the (approximate) efficient frontier, then maximize the index of satisfaction on this frontier. The one-step mean-variance approximation, *i.e.*, fixing the “Lagrange multiplier” λ , is only valid under very strong assumptions, *e.g.*, gaussian prices and exponential utility.

The following problems are usually considered equivalent, but this is only the case for affine constraints:

$$\begin{aligned} &\text{Maximize } E[\Psi_\alpha] \text{ such that } \text{Var} \Psi_\alpha = v \\ &\text{Maximize } E[\Psi_\alpha] \text{ such that } \text{Var} \Psi_\alpha \leq v \\ &\text{Minimize } \text{Var} \Psi_\alpha \text{ such that } E[\Psi_\alpha] \geq e. \end{aligned}$$

Chapter 7: Bayesian estimators

Bayesian statistics differs from classical inference in two regards:

- we provide some *prior* information as input;
- the output is not a single number or vector but a whole distribution – the *posterior*.

A *classical-equivalent* estimator is the single number (or vector) obtained as a location parameter of the posterior distribution – *e.g.*, *maximum a posteriori* (MAP) estimators, Bayes–Stein estimators, shrinkage estimators.

For some prior and model distributions (called *conjugate distributions*), the posterior is computable in closed form; for instance, the *normal inverse Wishart* distribution specifies the joint distribution of (μ, Σ) as

$$\begin{aligned} \mu | \Sigma &\sim N\left(\mu_0, \frac{\Sigma}{T_0}\right) \\ \Sigma^{-1} &\sim W\left(\nu_0, \frac{\Sigma_0^{-1}}{\nu_0}\right) \end{aligned}$$

and the data is

$$X_t | \mu, \Sigma \sim N(\mu, \Sigma).$$

Similar computations can be performed with factor models.

The prior distribution (or its location parameter) can be defined by

- inverting the unconstrained allocation function $\theta \mapsto \alpha(\theta) = \text{Argmax} S_\theta(\alpha)$, *i.e.*, finding the θ for which the market weights maximize the satisfaction index (the Black–Litterman prior is of this kind) – you might want to add a few constraints, though;

- a (constrained) maximum likelihood estimator.

Chapter 8: Evaluating allocations under uncertainty

An *allocation* is not a single set of weights but a function (a random variable)

$$\text{available information} \mapsto \text{weights}.$$

The *cost of randomness* is the difference between the satisfaction of the best portfolio given perfect insight, *i.e.*, the portfolio with the highest ex-post returns (it is likely to contain a single security) and the optimal (diversified) portfolio.

The *opportunity cost* (OC) is the difference between the satisfaction of the optimal allocation (assuming perfect knowledge of the market distribution) and that of the allocation actually chosen; constraint violations should be expressed in monetary terms (often in an ad hoc way).

Cost of randomness and opportunity cost are random variables.

Prior allocation, *i.e.*, allocation that does not use the information available, is the analogue of a fixed estimator: it is extremely biased.

Sample-based allocations are not too biased but have a very scattered opportunity cost (they are inefficient): the optimal allocation function is very sensitive to its inputs and *leverages* estimation error.

Chapter 9: Shrinkage allocation decisions

Bayesian allocation maximizes the expected utility of the investor’s objective, but the expectation is computed with respect to the posterior distribution; this is the analogue of a classical-equivalent bayesian estimator and the opportunity costs are less scattered than with sample-based allocation. Bayesian methods are *non-linear shrinkage* methods.

Black–Litterman allocation shrinks, not the market parameters, but the market distribution, towards the investor’s prior. The investor provides a random variable V that depends on the as-yet unknown market invariants X , for instance, $V \sim N(w'X, \phi^2)$, where X are the market returns, w' a portfolio on which the investor has a view and ϕ^2 the confidence of this view. Given a realization v of V and the knowledge of the distribution of $V|X$, we can compute the Black–Litterman distribution $X|V = v$ and then the corresponding Black–Litterman allocation decision. The prior X need not be the market invariants, but can be an “official” model. In the gaussian, linear case, the computations are straightforward.

The Mahalanobis distance between the market expectations μ and the Black–Litterman expectations μ_{BL} follows a χ^2 distribution, which can be turned into a p -value, to spot views in contradiction with the prior. To identify which view is responsible, just differentiate this p -value with respect to each view.

Resampled allocation proceeds as follows:

- Estimate the market parameters $\hat{\theta}$ from the data;
- Create new samples, by parametric bootstrap;
- For each sample q , estimate the market parameters θ_q and the corresponding optimal allocation α_p ;
- Average those optimal allocations.

However, resampled allocation can violate investment constraints (e.g., the maximum number of securities) and are difficult to stress-test – see Scherer’s book for more arguments against it.

Robust allocation replaces point estimates of market parameters by *uncertainty regions*: $\text{Argmax}_{\alpha} S(\alpha, \theta)$ becomes

$$\text{Argmax}_{\alpha} \text{Min}_{\theta \in \Theta} S(\alpha, \theta).$$

This depends on the choice (size, shape) of the uncertainty regions; with elliptical uncertainty regions, the problem is a second order cone program (SOCP).

Since bayesian methods output a whole distribution instead of a single value, *robust bayesian allocation* uses the corresponding location-dispersion ellipsoids as uncertainty sets; the radius of the chosen ellipsoid is the investor’s aversion to estimation risk. Under gaussian assumptions, the robust, bayesian, (two-step) mean-variance framework is amenable to explicit computations.

***Beyond Black–Litterman:
views on non-normal markets***
**A. Meucci
Risk (2006)**

The Black–Litterman framework can be generalized to a non-gaussian framework using the *copula-opinion pooling* (COP) methodology:

- Choose a prior distribution (e.g., by an equilibrium argument, as in the traditional BL framework, or by backward-looking estimation, or with forward-looking market-implied values – the prior could be a company-wide model on which individual managers would add their views),

$$V \sim (f_M, F_M, \phi_M)$$

(f_M is the probability distribution function (pdf), F_M the cumulative distribution function (cdf), ϕ_M the characteristic function);

- Choose the views $V = PM$ (each row of P is a portfolio; those portfolios should be independent) and the (marginal) distribution of each of them,

$$\hat{V}_k \sim (f_{\hat{V}_k}, F_{\hat{V}_k}, \phi_{\hat{V}_k})$$

- Blend these distributions with those, ($f_{V_k}, F_{V_k}, \phi_{V_k}$) induced by the market (separately for each view):

$$\tilde{V}_k \sim f_{\tilde{V}_k} = (1 - c_k)f_{V_k} + c_k f_{\hat{V}_k}$$

- Compute the joint posterior distribution of the views, by combining the marginal posterior distribution with the copula induced by the market distribution;

- Complete P into a basis:

$$\begin{pmatrix} V \\ W \end{pmatrix} = \begin{pmatrix} P \\ P^\perp \end{pmatrix} M \quad \text{or} \quad M = \begin{pmatrix} P \\ P^\perp \end{pmatrix}^{-1} \begin{pmatrix} V \\ W \end{pmatrix}$$

and compute the posterior distribution

$$\tilde{M} = \begin{pmatrix} P \\ P^\perp \end{pmatrix}^{-1} \begin{pmatrix} \tilde{V} \\ W \end{pmatrix}.$$

The computations can be carried out explicitly with a skewed T prior and uniform views; the result can be used in an expected shortfall optimization.

Beyond Black–Litterman in practice
**A. Meucci
Risk (2006)**

The COP approach can be applied even if the prior is not amenable to computations, by representing it by a large number (100,000) of Monte Carlo simulations:

- Generate Monte-Carlo samples from the market prior: M ;
- Turn those samples into marginal view priors: $V = PM$;
- Compute the view posterior distributions, using the cdf of the investor’s views;
- Compute the market copula;
- Combine the view posterior marginal distributions with the market copula;
- Compute the market distribution:

$$M = \begin{pmatrix} P \\ P^\perp \end{pmatrix}^{-1} \begin{pmatrix} \tilde{V} \\ P^\perp M \end{pmatrix}.$$

Convex vs concave dynamic allocation
A. Meucci (2002)

In a 2-asset world (cash and a stock), the author compares

- a buy-and-hold strategy;
- a call option (or the corresponding replicating portfolio);
- portfolio insurance (CPPI – the article clearly explains what it is);
- a power-utility maximizing strategy.

Portfolio insurance and call options yield a very similar payoff (indeed, the strategies are almost identical), while the power-utility maximizing strategy has a concave payoff (you would prefer a convex one).

The plots of the payoff of those four strategies are impressive – if you want to promote CPPI or badmouth portfolio optimization (but the results are specific to power-utility).

Broadening horizons
**A. Meucci
Risk Magazine (2004)**

Care should be taken when estimating the market distribution at a short (1 month) horizon and using it

at a longer (several years) one: use the characteristic function. The article details this switch between logarithmic and linear returns in a gaussian world and applies the results to portfolio optimization.

Robust bayesian allocation
A. Meucci

Robust bayesian allocation yields an efficient frontier parametrized by 3 parameters: the target variance, the size of the confidence ellipsoid on the mean, the size of the confidence ellipsoid on the variance matrix – but the efficient frontier is actually 1-dimensional, those 3 parameters, mixing market risk and estimation risk, can be combined in a single parameter.

Bayesian diagnostics for portfolio allocation
F. Corielli and A. Meucci

To decide whether to or how much to rebalance your portfolio towards the optimal portfolio, do not compare the composition of the portfolios (this is not robust) but their utility.

This article actually compares the investor's views and the equilibrium portfolio, in the Black–Litterman framework, and gauges how far from the equilibrium one can go; this will be different for each view: check the derivative of the utility wrt the confidence in each view.

***A common pitfall
in mean-variance asset allocation***
**A. Meucci
Wilmott (2001)**

When using mean-variance asset allocation, investors often mix linear returns and log-returns: the risk matrix is estimated using log-returns (the variance of the ratio returns is then:

$$\mu_{\text{ratio}} = \exp(\mu + \frac{1}{2} \text{diag } \Sigma) - 1$$

$$\Sigma_{\text{ratio}} = \exp(\mu + \frac{1}{2} \text{diag } \Sigma)' (\exp \Sigma - 1) \exp(\mu + \frac{1}{2} \text{diag } \Sigma)$$

but people rarely bother computing it – furthermore, the mean-variance framework assumes that the ratio returns follow an elliptical distribution), the forecasted alphas are ratio returns but are projected to a longer horizon as if they were log-returns. In the short term, those mistakes are harmless, but lead to very different allocations for long-term strategies – the author suspiciously finds that long-term investors should prefer less volatile assets.

Assessing views
G. Fusai and A. Meucci
Risk (2003)

When using the Black–Litterman framework to blend a general model (usually, universal equilibrium) with views, you may want to check how far from the model those views are: this can be measured with the Mahalanobis distance between the model alpha and the

Black–Litterman alpha (they use a RiskMetrics variance matrix, *i.e.*, an exponentially-weighted sample non-central variance matrix). The distance can be brought back to $[0, 1]$ assuming a χ_N^2 distribution (N is the number of assets). If the distance is too large and has to be reduced, you can choose which view to alter by differentiating the distance with respect to the returns of the views and selecting the view with the highest derivative.

Pitfalls in linear models for style analysis
F. Corielli and A. Meucci
Statistical methods and applications (2004)

Debunking bad econometrics (a case study for an econometrics (OLS) course): it is usually a bad idea to consider returns and is preferable to use prices and numbers of shares; some models that are often assumed linear are actually not linear and should be tackled with non-linear regression. There is no loss of parsimony: this is simply “the model you had in mind”, with exactly the same parameters, instead of an approximation. Using non-linear regression might not be that good an idea if you want reliable estimates, but is preferable if you want to compare models.

***Anomalies in the foundations
of ridge regression***
D.R. Jensen and D.E. Ramirez
arXiv:math.ST/0703551

Many people claim, without proof, that *ridge regression*, *i.e.*, replacing the ordinary least squares (OLS) estimator

$$\hat{\beta} = (X'X)^{-1}X'Y$$

by the biased estimator

$$\hat{\beta} = (X'X + \lambda I)^{-1}X'Y$$

is equivalent to a constrained regression

$$\begin{aligned} &\text{Minimize } (Y - X\beta)'(Y - X\beta) \\ &\text{such that } \beta'\beta \leq c^2 \end{aligned}$$

or

$$\begin{aligned} &\text{Minimize } (Y - X\beta)'(Y - X\beta) \\ &\text{such that } \beta'\beta = c^2. \end{aligned}$$

This is actually wrong.

***Forecasting the CATS benchmark with the
double vector quantization method***
G. Simon et al.
arXiv:math.ST/0703143

Double vector quantization (DVQ) can be used to predict a time series (the data comes from the competition on artificial time series (CATS) of the IJCNN 2004 (international joint conference on neural networks)):

- The data is modelled as a non-linear auto-regressive (NAR) time series whose order is computed from the *correlation dimension*

$$p = 2D_{\text{cor}} + 1$$

$$D_{\text{cor}} = \lim_{r \rightarrow 0} \frac{\ln C(r)}{\ln r}$$

$$C(r) = \lim_{n \rightarrow \infty} \frac{2}{n(n-1)} \sum_{1 \leq t < t' \leq n} \mathbf{1}(\|x_t - x_{t'}\| < r);$$

$C(r)$ is the (normalized) number of points in a hypersphere of radius r centered on x_t ; D_{cor} can be estimated by plotting $C(r)$ versus r and looking at the slope of the curve in the middle of the plot;

- Compute a self-organizing map (SOM – any other *quantization* algorithm would do) for the $\mathbf{x}_t = (x_t, \dots, x_{t-p+1})$ and another one for the *deformation regressors* $\mathbf{y}_t = \mathbf{x}_t - \mathbf{x}_{t-1}$;
- Compute the transition probabilities from the cells of the first map to those of the second; use them to *simulate* (do not just take the most probable values) the missing values.

***Measuring financial contagion:
a copula approach***
J.C. Rodriguez

Journal of empirical finance (2007)

You can model pairs of exchange rates, around crises, using switching copulas:

- the marginals are SWARCH (switching ARCH) models;
- the copula is a mixture of Frank, Clayton and Gumbel copulas; the parameters do not depend on the state variable but the weights do.

The model is compared with a switching Student copula with SWARCH marginals using the AICC (the AIC corrected for small sample size; this is better than the BIC is you are more interested in model selection than actual forecasts). Tests are then performed on the upper (and lower) *tail dependence*:

$$\lambda_u = \lim_{u \rightarrow 1} P[Y > G^{-1}(u) | X > F^{-1}(u)]$$

$$= \lim_{u \rightarrow 1} \frac{1 - 2u + C(u, u)}{1 - u}$$

$$\lambda_l = \lim_{u \rightarrow 0} P[Y < G^{-1}(u) | X < F^{-1}(u)]$$

$$= \lim_{u \rightarrow 0} \frac{C(u, u)}{u}$$

where F and G are the cdf of X and Y and C their copula.

Asian currency crises exhibit tail dependance and asymetry, while latin American ones do not.

***Specification and estimation of discrete time
quadratic stochastic volatility models***
H. Kawakatsu
Journal of empirical finance (2007)

The log-quadratic stochastic volatility model is:

$$y_t = \sigma_t u_t$$

$$\ln \sigma_t^2 = a_0 + a_1 x_t + a_2 x_t^2$$

$$x_{t+1} = \phi x_t + \sqrt{1 - \phi^2} v_{t+1} \begin{pmatrix} u_t \\ v_{t+1} \end{pmatrix} \sim N \left(0, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right)$$

***A closed form approach to the valuation and
hedging of basket and spread options***
S. Borovkova et al.
Journal of derivatives (2007)

Option pricing often relies on a log-normal distribution. When the underlying is a linear combination of assets, sometimes with negative coefficients (*commodity basket options*, e.g., *spread options*: spark spread (difference between electricity and gas price), crack spread (difference between crude oil and refined products), soybean crush spread), this no longer works: instead, one can use a shifted and/or reflected log-normal distribution.

***An algorithm for simulating bermudan option
prices on simulated asset prices***
B.N. Hugel and N. Rom-Poulsen
Journal of derivatives (2007)

How to price options when the underlying (e.g., an already complicated option) has itself to be priced by Monte Carlo simulation – in particular, the simulated prices will be too volatile and this bias should be corrected.

Algebraic statistical models
M. Drton and S. Sullivant
arXiv:math.ST/0703609

An *algebraic exponential family* is a semi-algebraic subset (up to diffeomorphism) of a regular exponential family. It is “well-behaved”: its maximum likelihood estimators are asymptotically gaussian.

Using *semi-algebraic sets*, i.e., allowing for inequalities, has applications in phylogenetics with *conditional independance models*, e.g., $X \perp\!\!\!\perp Y | Z \vee U \perp\!\!\!\perp Z$.

The article is very abstract: for instance, regular algebraic families are defined in a non-constructive way, using the Laplace transform.

***Transitional densities of diffusion processes:
a new approach to solving
the Fokker–Planck equation***
A.S. Hurn et al.
Journal of derivatives (2007)

Numerically computing the transition probability distribution function (pdf) of a diffusion process via the Fokker–Planck PDE is tricky: the initial condition is a Dirac measure. Instead, one can consider the cumulative distribution function (cdf) and numerically differentiate it.

***The implied volatility term structure
of stock index options***

S. Mixon

Journal of empirical finance (2007)

$\left(\frac{\partial \text{implied volatility}}{\partial \text{maturity}}\right)_{\text{ATM}}$ can help predict future implied volatility.

When is inter-transaction time informative?

C. Furfine

Journal of empirical finance (2007)

Faster arriving trades move prices more than slower ones; the effect is weaker for actively traded stocks; faster arriving stocks in the same direction are more informative.

***The performance of hedge fund strategies and
the asymmetry of return distributions***

B. Ding and H.A. Shawky

European financial management

To study hedge funds, do not use returns but residual returns, *i.e.*, remove the effect of the market, the Fama-French factors and the square of the market (the corresponding coefficient of the regression is the *coskewness*: that added term yields a higher-moment CAPM).

***Risk measures for hedge funds:
a cross-sectional approach***

B. Liang and H. Park

European financial management

To measure risk, prefer expected shortfall (ES) or *tail risk* (TR) (*i.e.*, $E[(X - \bar{X})^2 | X < \text{VaR}]$) to value at risk or semi-variance.

The Cornish-Fisher expansion is preferable to a non-parametric estimation of ES or TR – the authors do not consider parametric (extreme value theory, EVT) estimation – the non-parametric approach should become preferable with more (daily) data.

***Hedge fund indices:
reconciling investability and representativity***

F. Goltz et al.

European financial management

It is possible to build investable and representative hedge fund indices (classical HF indices are not investable).

***Sources of contrarian profits
in the Japanese stock market***

P.-H. Chou et al.

Journal of empirical finance (2007)

Reversal (with short (1 month) or long (2 years) horizons) works in Japan, because of the *lead-lag* effect, *i.e.*, because of *cross-correlations* between the stock returns.

***Industry information diffusion and the
lead-lag effect in stock returns***

K. Hou (2007)

The *lead-lag* effect (past large-firm returns and future small-firm returns are related) is an intra-industry phenomenon, more pronounced in small (size, volume, market share), less competitive and neglected (analyst coverage, analyst dispersion) industries (institutional ownership also plays a role).

Portfolio selection with heavy tails

N. Hyung and C.G. de Vries

Journal of empirical finance (2007)

For portfolio construction, first order tail expansions are not precise enough: they lead to extreme allocations, with all the weight in the asset with the thinnest tail – in fact, the tail thickness of a portfolio is well approximated by that of its component with the fatest tail.

The article derives asymptotic expansions of $P(X_1 + X_2 > s)$ assuming that X_1 and X_2 are independent and satisfy

$$P(X > s) = As^{-\alpha} (1 + Bs^{-\beta} + o(s^{-\beta})).$$

Are IPOs really overpriced?

S.X. Zheng

Journal of empirical finance (2007)

Studies claiming that IPOs are overpriced often overlook their growth potential (omitted variable problem).

***Interaction of stock return momentum
with earnings measures***

I. Figelman

Financial analysts journal (2007)

Among companies with poor past returns, companies with high ROE or low quality (accruals) underperform.

Migration

E.F. Fama and K.R. French

Financial analysts journal (2007)

The size premium can be explained by small caps becoming large caps and conversely. This remains valid for the value premium – but these are ex post explanations.

***Bayesian learning in financial markets:
testing the relevance of information precision
and price discovery***

N. Hautsch and D. Hess

**Journal of financial and quantitative analysis
(2007)**

Investors are not blind to heteroskedasticity: use a “quality” factor in your models.

Small-world MCMC and convergence to multi-modal distributions: from slow-mixing to fast mixing

Y. Guan and S.M. Krone
arXiv:math.PR/0703021

To reduce mixing time in an MCMC (Markov Chain Monte Carlo) simulation on a multimodal space, you can use, as a proposal distribution, a mixture of a thin-tailed and a fat-tailed distribution. This can be interpreted in graph-theoretic terms as a small-world effect – some people will also see a Levy process.

The article also contains a theoretical (functional analysis) presentation of continuous Markov chains.

A generative model for feedback networks
D.R. White et al. (2005)

There are many models of *random networks*, but not is satisfactory. Here is yet another one, that mimicks kinship or corporate alliances networks:

- Select a node i ,

$$P(i) \propto \text{degree}(i)^\alpha$$

- Select a distance d ,

$$P(d) \propto d^\beta$$

- Generate a search path, from i , of length d , iteratively, randomly choosing a neighbour l , without going through the same node twice,

$$P(l) \propto 1 + u(l)^\gamma$$

where $u(l)$ is the unused degree of l

- If this works (we do find a path), then we add an edge between i and the end of the path; otherwise, we add a new node and link it to i .

Times-series behavior of share repurchases and dividends

B.S. Lee and O.M. Rui
Journal of financial and quantitative analysis
(2007)

The authors compare two hypotheses to explain the fall in the number of companies paying dividends and the rise of buy-backs: either taxes or temporary cash flows – the latter is more likely.

Giving content to investor sentiment: the role of media in the stock market

P.C. Tetlock
Journal of Finance (2007)

The author processes a daily Wall Street Journal editorial over 16 years through a *content analysis* software (General Inquirer, non-free, non-commercial – check <http://www.wjh.harvard.edu/~inquirer/inqtabs.txt> and <http://textanalysis.info>), counting the words in 100 predefined categories;

a principal component analysis (PCA) of the result yields a *pessimism measure*.

It predicts volume, volatility, depth, but not returns – though it is correlated with downwards pressure followed by a reversal.

An examination of alternative portfolio rebalancing strategies applied to sector funds
Journal of asset management (2007)

You should rebalance your portfolio (to make it equal-weighted), if only to limit the effect of bubbles, but there are no significant differences between rebalancing on a trigger or at regular intervals (once a year on average for a portfolio made of index funds).

Event studies with a contaminated estimation period
N. Aktas et al.
Journal of corporate finance (2007)

In event studies, the learning period can be contaminated, *i.e.*, can already contain events (e.g., if you are looking for potential M&A bidders). This can be taken into account with *Markov switching models*, distinguishing between high and low periods.

The article also reviews the tests classically used in event studies.

All events induce variance: analyzing abnormal returns when effects vary across firms
S.E. Harrington and D.G. Shrider
Journal of financial and quantitative analysis
(2007)

Beware of heteroskedasticity in event studies – prefer robust tests.

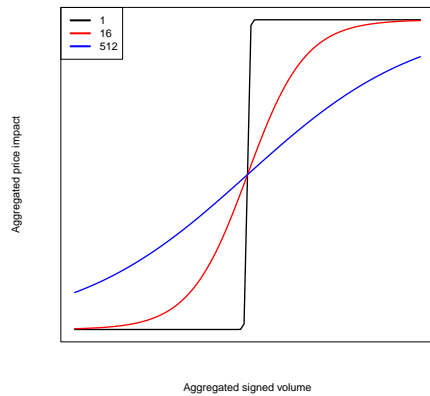
A theory for long-memory in supply and demand
F. Lillo et al. (2005)

The long memory of the sign of trades can be explained by *market clearing* and hidden orders.

Statistical mechanics of liquidity providers and liquidity takers
J.D. Farmer
Barclays Capital (2007)

The *market impact* is not a “second order effect” as some claim, but the main price formation mechanism (buyer-initiated trades raise the price, seller-initiated ones lower it).

The statistical properties of the price impact can be examined as follows: plot the price impact (logarithm of the ratio of prices) aggregated over 2 (resp. 4, 8, 16, ..., 1024) trades versus the corresponding aggregated signed volume. After rescaling, you get:



This shape can be explained by the following model:

- the volume follows a power law: $p(v) = 1/v^{\alpha+1}$;
- the impact is $f(v) = \text{sign } v \cdot |v|^\beta$

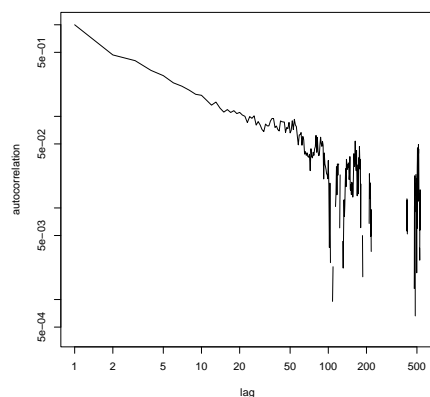
which also gives the slope of the limiting curve:

$$\frac{\text{aggregated returns}}{\text{aggregated signed}} \sim N^{-\kappa}$$

where N is the number of aggregated transactions and κ depends on α and β .

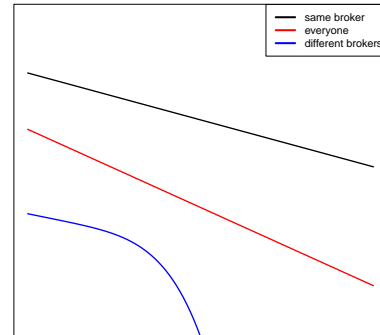
However, this does not give the whole picture: it does not account for the following two facts:

- the orders that arrive are not iid;
- the volume sign has *long-memory*: if you plot the log autocorrelation versus the log of the lag, you get a line, *i.e.*, the autocorrelation decreases very slowly: even after two weeks (remember, these are high-frequency data), autocorrelation is still statistically significant. (As far as statistical estimation is concerned, this is bad news: the error of most estimators decrease as $1/N^{1-H}$, where H is the *Hurst exponent* (expect it to be around 3/4), instead of the usual $1/\sqrt{N}$.)



The main cause of this long memory seems to be, not herding, but the slicing of large trades into smaller ones: indeed, the autocorrelation of the volume signs is more persistent if you restrict the data to a single

broker and the long memory disappears if you consider different brokers.



Long memory does not contradict market efficiency thanks to the following (equivalent) phenomena:

- the price impact is temporary and cancels the long memory;
- there is liquidity imbalance.

One can model the non-iid nature of the orders by assuming that:

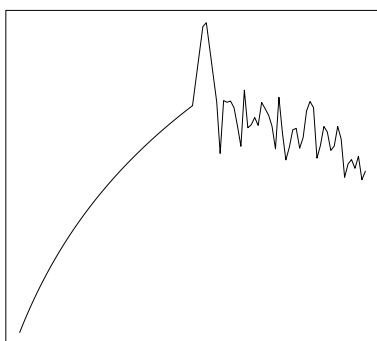
- large, *hidden orders* are split into smaller ones;
- large orders follow a power law;
- the market participants know when a large order starts but not when it stops – because of the power law, the probability that the next order will be of the same sign increases with the size of the hidden trade);
- the market is efficient.

This suffices to show that

$$\text{impact} \sim \log N$$

where N is the number of chunks (their size is fixed). This is confirmed by the data – on the other hand, the hypothesis that order flows are anonymized (market participants do not know when a hidden trade starts) do not fit the data.

The life of a hidden order looks like this (the model also predicts clustered volatility; it can take more than four transactions for the market participants to realize that the hidden trade has ended):



In a nutshell:

- power laws are everywhere
- they can be explained by the slicing of large (hidden) orders
- the market participants can recognize those hidden orders, but they do not immediately notice when they have stopped
- the logarithm law, $\text{impact} \sim \log N$ suggests that the *capacity* of your strategy could be higher than you think.

The speaker likens the financial world to an *ecosystem*, with *competition* (if A grows bigger then B loses money; if B grows bigger, then A loses money) and *prey/predator* relations (if A grows bigger, then B earns money; if B grow bigger, then A loses money) – you can also imagine relations not encountered in wildlife ecosystems.

The key role of liquidity fluctuations in determining large price changes

F. Lillo and J. D. Farmer

Fluctuation and noise letters (2005)

In a *double continuous auction market*, such as the LSE (London stock exchange) or the NYSE (New York Stock Exchange), impatient traders send *market orders*, to be executed immediately, at the best possible price, while patient investors send *limit orders* or *quotes*, to be executed at a given price, which are stored in a (publicly visible) queue, the *limit order book*. (At the beginning and the end of the day, there is also an auction, where everyone sends quotes for 10 minutes, and they are only matched at the end. And this is only the *on-book market* (SETS in London, downstairs market in New York): there is an *off-book market* (SEAQ, upstairs market), where trades are decided bilaterally, on the phone, and only made public afterwards.)

Large price changes are determined by holes in the order book and can be triggered by small volumes. Not only the first *gap* (the closest to the best price) matters, but also the others – *sparse* order books call for large price changes.

(The gap size or the sparsity of the book can be measured from the tick data we already have.)

There's more to volatility than volume

L. Guillemot et al. (2005)

Volatility clustering cannot be explained by a change-of-time either using the volume clock or the number-of-transactions clock.

An empirical behavioral model of price formation

S. Mike and J. D. Farmer (2005)

An *agent-based model* of price formation, comprised of

- An order placement model, using a Student T distribution with fewer than 2 degrees of freedom;
- A cancellation model, taking order imbalance and the position wrt the best price

can reproduce the observed distributions of returns and bid-ask spreads.

International bond market cointegration using regime switching techniques

A. Davies

Journal of fixed income (2007)

There is *cointegration* (“integration”) between national bond markets; the common trend presents structural changes.

Simple forecasts and paradigm shifts

H. Hong et al.

Journal of finance (2007)

The following agent-base model:

- A and B are two independant AR processes (think of A as value and B as growth);
- (Log-)dividend payments are given by a linear regression: $D_{t+1} = A_t + B_t + \text{noise}$;
- Investors choose between the two incorrect models $D_{t+1} = A_t + \text{noise}$ and $D_{t+1} = B_t + \text{noise}$; they change their model when the difference in log-likelihood exceeds a certain threshold (this is the confidence level of the statistical (LR) test); that threshold varies from investor to investor;
- The combined behaviour of those investors drives the prices

explains the stylized facts:

- book-to-market effect;
- volatility and skewness are stochastic.

How to make better choices

K. Douglas and D. Jones

New Scientist 2602, May 2007, p.35–43

This article lists several *behavioural biases* that could be applied to bahavioural finance:

- loss aversion
- information overload; to many choices (e.g., in a supermarket, when faced with two queues, you have a 50% chance of chosing the fastest one; with 100 queues, this probability drops to 1%)

- emotions (anger or disgust are bad, depression is good)
- confirmation bias
- anchoring effect (*i.e.*, bayesian shrinkage with a meaningless prior)
- sunk cost
- framing effect (the half full/empty glass)
- social (or peer) pressure.

See also: *Intuition, its powers and perils* (D.G. Myers), *The paradox of choice* (B. Schwartz), *Why choose this book?* (R. Montague), *Why smart people make big money mistakes and how to correct them* (G. Belsky and T. Gilovich).

A tale of two anomalies: the implication of investor attention for price and earnings momentum
K. Hou et al. (2007)

This article examines interactions between:

- investor *attention* (they use the volume as a proxy for it; one could also use analysts' coverage): excessive attention can lead to over-confidence and explain price momentum while lack of attention could lead to post earnings announcement drift (PEAD);
- market direction
- price momentum
- earnings momentum (*standardized unexpected earnings* or SUE).

For instance: price momentum is profitable in up markets; earnings momentum is profitable in down markets.

Liquidity aggregation: what institutional investors need to know
G. Butler
Algorithmic trading (2007)

Presentation of *alternative trading systems* (ATS), *i.e.*, brokers' internal crossing engines, *ATS aggregators* and *electronic communication networks* (ECN) (network between the ATS).

Shedding light on Dark liquidity
J. Suryawanshi and J. Fox (2007)
Algorithmic trading (2007)

(Empty article)

More and more shares are crossed outside the displayed markets (ATS, ECN).

A boosting approach for automated trading
G. Creamer and Y. Freund
Algorithmic trading (2007)
Journal of trading (2007)

The authors implemented the following strategy for the *Penn-Lehman automated trading* (PLAT) competition (a competitor of the university of Michigan's *trading agent competition* (TAC)), that simulates ISLAND, one of the majors ECNs:

- Using Rmetrics, compute technical indicators over the past 90 days (they are detailed in an appendix);
- Apply *boosting* on the trading rules (*decision trees*) derived from those indicators;
- Use a constantly rebalanced portfolio (50% cash, 50% long; or 125% cash, 25% short); peek into the *limit order* book and add limit orders just beyond the best ones; only use *market orders* if this fails.

Stupid data miner tricks: overfitting the S&P 500
D.J. Leinweber
Journal of investing (2007)

Data mining, *i.e.*, the extraction of patterns from large data sets, can go awry and become *data snooping*. To prevent this, the authors suggest to use:

- temporal and cross-sectional out-of-sample;
- a measure of data snooping, such as *White's reality check*;
- random data.

This sounds outdated: no mention of the bootstrap as an alternative to the in- and out-of-sample splitting; no mention of sensitivity measures (influence function, etc.) to measure how dependant on data variations the results are or robust methods to reduce this dependence; no mention of procedures that can cope with an excess of predictive variables (shrinkage, support vector machines (SVM), etc.).

Value and growth, theory and practice
J.S. Brush
Journal of portfolio management (spring 2007)

Value (V) and growth (ΔV) are two (negatively correlated but) different signals, measuring *static* and *dynamic* value. Growth beats value in the short term, while value persists for years. Thanks to the negative correlation, combining them lowers volatility (plot various blends of value and growth in the returns×volatility plane).

The author uses the following value factors:

- dividend to price;
- earnings to price;
- cash flow to price;
- expected earnings to price;
- book value to price

and the following growth factors:

- short-term change in earnings-to-price
- long-term change in earnings-to-price
- estimate revisions (number of upwards revisions, minus number of downwards ones, divided by the total number of revisions, time-weighted);
- earnings surprises (reported earnings minus consensus predicted earnings, at the time of the last earnings report);
- price momentum.

***Should owners of Nasdaq stocks
fear short-selling?***

S.E. Christophe et al.

Journal of portfolio management (spring 2007)

Short selling is not detrimental to stock returns:

- only extreme short selling (over 10%) has a negative impact on stock returns, but only 5 basis points per percentage point;
- dealer short-selling has a positive impact on returns: market makers only short when they receive many buy orders.

The article did not study the effect of short-selling on volatility or during a crash.

***Analysis of the interest rate sensitivity
of common stocks: empirical equity
duration is a useful measure***

F.K. Reilly et al.

Journal of portfolio management (spring 2007)

Equity duration (you might already be familiar with the notion from fixed income), defined from a regression

$$\Delta \text{price} \sim \Delta \text{IR}$$

or

$$\Delta \text{price} \sim \text{market returns} + \Delta \text{IR}$$

is very unstable and depends on the industry and the period (e.g., it changed in 2001).

Clarifications on “Beyond Markowitz”

A.B. Chhabra

Journal of wealth management (summer 2007)

Discussions on what to do with investors who have several goals, do not understand mean-variance optimality and prefer buckets: WAF (wealth allocation framework), DWH (discretionary wealth hypothesis) or HOG (holistic goal optimization).

***How to generate random matrices
from the classical compact groups***

F. Mezzadri

Notices of the AMS, May 2007

To get a random unitary matrix (since $U(N)$ is a compact group, we use its Haar measure), one could think of using the *Gram–Schmidt orthonormalization* – but it is numerically unstable. Instead, one can use the *QR decomposition*, accounting for its non-uniqueness, as follows:

- Take a random, gaussian, complex matrix Z (this probability space is called the *Ginibre ensemble*);
- Compute the *QR decomposition*: $Z = QR$;
- Set $Q' = Q \cdot \text{diag} \left(\frac{r_{11}}{|r_{11}|}, \dots, \frac{r_{NN}}{|r_{NN}|} \right)$

The article gives actual code (in Python, using *scipy*) and examples from applications of random matrix theory (RMT) in quantum mechanics, approximating infinite-dimensional operators with large matrices.

***Enhanced index investing
based on goal programming***

L.-C. Wu et al

Journal of portfolio management (2007)

Goal programming can be used to tackle dual-objective problems, such as active management (*i.e.*, follow the market and beat it, *i.e.*, have a low tracking error with the market and high expected returns): just set a goal for each objective (say, 3% tracking error and 7% returns) and minimize the sum of the distances to those goals.

The authors fail to see that the objectives should be expressed in the same (monetary) unit, lest the user want to tweak the weight given to each.

***Robust portfolio optimization:
recent trends and future directions***

F.J. Fabozzi et al.

Journal of portfolio management (2007)

The article is an informal presentation of robust optimization – or an advert for the authors’ forthcoming book.

Robust optimization provides more stable weights and a better use of the turnover budget than classical mean-variance optimization.

Robust optimization is only equivalent to shrinkage towards the minimum variance portfolio in some very special cases (no uncertainty on the variance matrix, stock variance and estimation variance are proportional, no constraints).

Robust portfolio optimization can also be used to find a portfolio “statistically equivalent” to the optimal one that minimizes transaction costs.

Robust portfolio optimization is essentially a worst-case optimization: *zero net alpha adjustment* changes the shape of the uncertainty set to reflect the fact that estimation errors are not systematically in the wrong direction.

Robust *multi-period* problems can surprisingly prove more tractable than their non-robust analogues.

Robust optimization is not limited to portfolio construction, but also has applications in optimal trading, optimal hedging, econometric model estimation, model selection, etc.

***Does size matter? Assets under management
a questionable question***

G.C. Allen

Journal of portfolio management (2007)

Funds with large AUM (assets under management) are less risky but have lower returns; the effect is less marked with large capitalizations.

**Optimal execution for portfolio transitions:
minimizing opportunity cost**

M. Kritzman et al.

Journal of portfolio management (2007)

To find the best trade order for a *portfolio transition* (i.e., a drastic rebalance), look at the partial derivative of the tracking error (TE) (the distance between the current and desired portfolios) and choose trades that reduce the TE most. (Generally, people use any order that preserves sector neutrality.) You can also add *market impact*, proportional to

$$\sqrt{\frac{\text{trade size}}{\text{average daily volume}}}$$

to the TE.

**Measuring portfolio performance
using a modified measure of risk**

C. Adcock

Journal of asset management (2007)

The *Rubinstein-Leland beta*, defined as

$$\beta^{\text{RL}} = \frac{\text{Cov}(r, -(1+r_m)^{-b})}{\text{Cov}(r_m, -(1+r_m)^{-b})}$$

(where r are the portfolio ratio-returns, r_m the market ratio-returns, the investor's utility is $U(x) \propto x^{1-b}/(1-b)$ and b can be estimated as $b = \frac{1}{2} + (E \ln(1+r_m) - \log(1+r_f))/\sigma_m^2$) does not seem to add anything to the CAPM beta: this can be proven for elliptical distributions, and empirically tested for asymmetric ones.

**Can robust portfolio optimization help
to build better portfolios?**

B. Scherer

Journal of asset management (2007)

Robust optimization is no better than bayesian shrinkage – it actually is a shrinkage towards the minimum variance portfolio, with a difficult-to-interpret-and-predict coefficient. Furthermore, it decouples model uncertainty and risk, while they are inseparable.

(“Robust optimization” is actually a misnomer: it should be called “worst case optimization”.)

Attribution:

Modeling asset characteristics as portfolios

R. Grinold

Journal of portfolio management (2006)

The author explains how to analyze a portfolio using only covariances between portfolio returns.

Let X be the random variable of stock returns, with $EX = \alpha$, $\text{Var } X = V$; let p be the portfolio weights and $P = p'X$ its returns. The ideal portfolio has weights $q = \lambda^{-1}V^{-1}\alpha$ and returns $Q = q'X$, where λ is the

risk aversion. One then has:

$$\alpha_P = \lambda \text{Cov}(P, Q)$$

$$\text{IR}_P = \frac{\alpha}{\text{Sd } P} = \lambda \text{Sd}(Q) \text{Cor}(P, Q)$$

$$\text{TC}_P = \frac{\text{IR}_P}{\text{IR}_Q} = \text{Cor}(P, Q)$$

$$U_Q - U_P = \frac{\lambda}{2} \text{Var}(Q - P)$$

(The transfer coefficient (TC) is a measure of implementation efficiency.)

By decomposing the portfolio weights into several explanatory portfolios (it is up to you to define them),

$$p = \sum_j \beta_j p_j + \varepsilon$$

(in case of multicollinearity problems, you might want to use some shrinkage; also not the absence of intercept: if some relevant factors are missing, they will be spread out over all the terms instead of being put in the residual portfolio), you can decompose the alpha (it is linear in p) into contributions for each source.

For an ex post analysis, just replace the alphas by realized returns.

The article suggests a report as follows:

- Portfolio IR: $\lambda \text{Sd } Q \text{Cor}(P, Q)$
- Portfolio risk: $\text{Sd } P$
- Portfolio alpha: $\lambda \text{Cov}(P, Q)$
- Portfolio TC: $\text{Cor}(P, Q)$
- For each explanatory portfolio:
 - Exposure: $\psi_j = \beta_j \frac{\text{Sd } P_j}{\text{Sd } P}$
 - Risk: $\psi_j \text{Cor}(P_j, P)$?
 - TC: $\text{Cor}(P_j, Q)$
 - IR Sd PTC_j
 - Contribution: $\psi_j \cdot \text{IR} \cdot \text{Sd } P \cdot \text{TC}_j$
- Residual exposure: $\text{Cor}(P, E)$
- Residual risk: $\text{Cor}(E, P)^2$?
- TC_{residual}: $\text{Cor}(E, Q)$
- Residual alpha: $\text{IR} \cdot \text{Sd } P \cdot \text{Cor}(P, E) \cdot \text{Cor}(E, Q)$

**Portfolio constraints and the fundamental law
of active management**

R. Clarke et al. (2002)

(Another article on the transfer coefficient.)

Maximize the Sharpe ratio

Axioma Advisor Newsletter, March 2007

The problem

$$\text{Maximize } \frac{w' \alpha}{\sqrt{w' Q w}}$$

$$\text{such that } w' 1 = 1$$

$$w \geq 1$$

$$\forall i \quad l_i \leq w' s_i \leq u_i$$

can be replaced by

$$\begin{aligned} & \text{Minimize } w'Qw \\ & \text{such that } w'\alpha = \bar{\alpha} \\ & w \geq 1 \\ & \forall i \quad \tau l_i \leq w's_i \leq \tau u_i \end{aligned}$$

The unknowns are w and τ and the desired weights are w/τ , but since you do not know $\bar{\alpha}$, you will have to perform several iterations to fine-tune it. This does not easily generalize to more complicated constraints or penalties (transaction costs, etc.)

Database cracking
S. Idreos et al
CIDR 2007

Database optimizations are usually done by means of indices or partitioning (*i.e.*, prescribing the order in which the data should be stored on the disk): this article suggests that the DBMS should dynamically reorder the data on disk in order to speed queries up.

This is already implemented in MonetDB, a (mainly) main-memory column-oriented DBMS, with SQL and XQuery interfaces.

An anthological review
of research utilizing MontyLingua:
a Python-based end-to-end text processor
M.H.T. Ling
The Python Journal

There are two natural language processing (NLP) Python libraries: NLTK (targeted at NLP teaching) and MontyLingua. This article reviews 19 articles using the latter.

Econometrics
B. Hansen (2000–2007)

There are several differences between econometrics and mainstream statistics:

- The use of *observational data* means that one cannot redo an experiment after changing something: *causality* cannot be inferred;
- Econometricians make as few hypotheses as possible, in particular, they rarely assume that the distributions are gaussian (or even known) or that the variables of interest are independant;
- They do not only want forecasts from their models, they also want inferences on the parameters of those models.

One can consider several *regression models*:

- *Linear regression*:

$$\begin{aligned} y &= x'\beta + e \\ E[e|x] &= 0 \end{aligned}$$

- *Homoskedastic linear regression*:

$$\begin{aligned} y &= x'\beta + e \\ E[e|x] &= 0 \\ E[e^2|x] &= \sigma^2 \end{aligned}$$

- *Linear projection* (we assume that the formula for the coefficient of the regression is valid at the population level and we try to see what we can say from a sample):

$$\begin{aligned} y &= x'\beta + e \\ E[e|x] &= 0 \\ \beta &= E[xx']^{-1}E[xy] \end{aligned}$$

- Homoskedastic linear projection;
- Gaussian linear regression.

The following estimators coincide:

- Take the formula for the linear projection model and replace expectations by means (this is called a *moment*);
- Minimize $\|y - x'\beta\|^2$: this is the *ordinary least squares* (OLS or LS) estimator;
- Assume that the predictors x and the noise e are independant, that the noise is iid gaussian and maximize the likelihood: this is the *maximum likelihood estimator* (MLE);
- Assume that (x, y) only take a finite number of values and compute the corresponding (multinomial) MLE;
- Maximize the R^2 (not mentioned in this book).

In the linear projection model, the LS estimator is the *best linear unbiased estimator* (BLUE) and *semi-parametric efficient* (*i.e.*, it has the smallest mean square error (MSE) among the feasible estimators; a *feasible* estimator is an estimator that does not use parameters we do not know, e.g., it cannot use σ).

The regression can be done in two steps:

$$\text{res}(y \sim x_1 + x_2) = \text{res}(\text{res}(y \sim x_2) \sim \text{res}(x_1 \sim x_2)).$$

In case of *multicollinearity*, *i.e.*, redundant predictive variables, it will be difficult to disentangle the effects of the predictors.

To measure the influence of isolated observations, you can compute their *leverage*, the *leave-out-one* (LOO) $\hat{\beta}$ and residuals.

The quality of a regression model can be measured by estimators of $\rho^2 = 1 - \sigma^2/\sigma_y^2$:

$$\begin{aligned} R^2 &= 1 - \frac{\hat{\sigma}^2}{\hat{\sigma}_y^2} \\ R_{\text{adj}}^2 &= 1 - \frac{s^2}{\hat{s}_y^2} \end{aligned}$$

where

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n} \sum \hat{e}_i^2 \\ s^2 &= \frac{1}{n-k} \sum \hat{e}_i^2 \\ \hat{\sigma}_y^2 &= \frac{1}{n} \sum (y_i - \bar{y})^2 \\ \hat{s}_y^2 &= \frac{1}{n-1} \sum (y_i - \bar{y})^2\end{aligned}$$

The least squares estimator is *consistent*: it converges, in probability, to the true value of the parameter. More precisely, $\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{d} N(0, V)$ where

$$\begin{aligned}Q &= E[xx'] \\ \Omega &= E[(x_i e_i)(x_i e_i)'] = E[x_i x_i' e_i^2] \\ V &= Q^{-1} \Omega Q^{-1}.\end{aligned}$$

For the homoskedastic projection model, the variance is $V^0 = Q^{-1} \sigma^2$.

Since the true value of V is not knowable, we estimate it:

- Traditional: $\hat{V}^0 = \hat{Q}^{-1} s^2$;
- Heteroskedasticity-robust covariance matrix:

$$\hat{V} = \hat{Q}^{-1} \hat{\Omega} \hat{Q}^{-1};$$

- MacKinnon–White jackknife estimator:

$$\hat{V}^* = (n-1) \sum (\hat{\beta}_{(-i)} - \bar{\beta})(\hat{\beta}_{(-i)} - \bar{\beta})';$$

- Andrew’s cross-validation estimator.

The *t*-statistic

$$t_n(\theta) = \frac{\hat{\theta} - \theta}{s(\hat{\theta})},$$

with $\theta = h(\beta)$ some function of the parameters,

$$\begin{aligned}s(\hat{\theta}) &= n^{-1/2} \sqrt{\hat{H}'_{\beta} \hat{V} \hat{H}_{\beta}} \\ H_{\beta} &= \frac{\partial h}{\partial \beta}(\beta),\end{aligned}$$

converges in distribution to $N(0, 1)$ (it is asymptotically *pivotal*, *i.e.*, the limit distribution no longer depends on the parameters) and can be used for (asymptotic tests) or (asymptotic) confidence intervals.

The *Wald test* is a higher-dimensional analogue:

$$W_n = n(h(\beta) - \theta_0)(\hat{H}'_{\beta} \hat{V} \hat{H}_{\beta})^{-1}(h(\beta) - \theta_0)';$$

it converges in distribution to χ_q^2 where $q = \text{rank } H_{\beta}$.

The *F*-test is a special test of the Wald test where one checks if one or several of the parameters is zero.

With gaussian linear model, one knows the distribution of the *T*-statistic and can replace the Wald test by a more precise *likelihood ratio* (LR) test – the former is a first-order approximation of the latter.

The *T*- and Wald tests do not work well with non-linear hypotheses – in that case (if you cannot transform the

hypothesis into a linear one), or if the sample is small prefer Monte Carlo simulations,

OLS is efficient in the projection model (and in the gaussian one, of course), but not in the linear regression model. However, the *generalized least squares* (GLS) estimator is efficient:

$$\begin{aligned}\tilde{\beta} &= (X' D^{-1} X)^{-1} (X' D^{-1} y) \\ D &= \text{diag}(\sigma_1^2, \dots, \sigma_n^2) \\ \sigma_i^2 &= \sigma^2(x_i) = E[e_i^2 | x_i].\end{aligned}$$

If you want a feasible estimator, replace σ_i^2 by some estimator $\hat{\sigma}_i^2$, *e.g.*, from a *skedastic regression* $\sigma_i^2 = \alpha' z_i$.

GLS is less robust than OLS: only use it if you really, really need it.

Heteroskedasticity can be tested with a Wald test on the skedastic regression; the White and Breusch–Pagan tests actually test for more: second-order independence (the z_i include the x_i , their squares and cross-products).

I have not read section 5.3 on forecast intervals.

Non-linear least squares estimate a (prescribed) non-linear expression for the conditional mean, still minimizing the sum of squared errors. Non-linear models need not be *identifiable*: *e.g.*, if $y = \beta x^{\gamma}$ and $\beta = 0$, then γ is not identified.

Least absolute deviations (LAD) minimizes the sum of the absolute values of the errors:

$$\theta_0 = \underset{\theta}{\text{argmin}} E |y - \theta|;$$

equivalently:

$$P(y \leq \theta_0) = P(y \geq \theta_0) = \frac{1}{2}$$

or

$$E[\text{sgn}(y - \theta_0)] = 0.$$

LAD works better when there are many observations bear the median, *i.e.*, when $f(0)$ is large. This is the case with many skewed distributions (there are more observations around the median than around the mean) and fat-tailed distributions (they are “pointy”).

LAD is a special case of *quantile regression*, usually implemented as a linear problem.

To test for non-linearity, just add non-linear functions of the regressors and perform a Wald test: the *RESET test* uses $\hat{y}^2, \hat{y}^3, \dots, \hat{y}^m$ and is particularly powerful at detecting *single-index models*: $y = G(x' \beta) + e$.

Omission of a relevant variable leads to the *omitted variable bias*. When the error is conditionally heteroskedastic, $E[e_i^2 | x_i] = \sigma^2$, the inclusion of irrelevant correlated variables reduces efficiency; without homoskedasticity, there is no clear rule.

Statistical tests do not help for *model selection* (they are not symmetric); AIC (an estimator of the Kullback–Leibler distance) selection is inconsistent (it overfits): prefer the BIC.

A statistic can be written as $T_n(F)$ where F is the (population) distribution function of the data and n the number of observations. To estimate the cumulative distribution function (cdf) $G_n(u, F) = P(T_n \geq u | F)$, the *non-parametric bootstrap* makes two approximations:

- Use a consistent estimator of the cdf F , such as the *empirical distribution function* F_n ;
- Use a Monte Carlo simulation to estimate $G_n^*(u) = G_n(u, F_n)$ – in other words, we sample *with replacement* (to simulate the multinomial distribution F_n) from the sample.

The *bias-corrected* bootstrap estimator is $\hat{\theta} - (\bar{\hat{\theta}}^* - \hat{\theta}) = 2\hat{\theta} - \bar{\hat{\theta}}^*$ where $\hat{\theta}$ is the value of the estimator T on the sample, $\hat{\theta}^*$ is the value of T on a bootstrap sample, $\bar{\hat{\theta}}^*$ is the average value value of T on the bootstrap samples, $(\bar{\hat{\theta}}^* - \hat{\theta})$ is an estimator of the bias.

Variance estimators are not very useful for the bootstrap: prefer *confidence intervals*:

- The oft-used *Efron percentile interval*, $[\hat{\theta} - q(\alpha/2), \hat{\theta} - q(1 - \alpha/2)]$, where q are the quantiles of $\hat{\theta}^* - \hat{\theta}$, is biased (unless the distribution of symmetric) and should be avoided;
- The *percentile interval*, $[\hat{\theta} - q(1 - \alpha/2), \hat{\theta} - q(\alpha/2)]$, is preferable;
- The *percentile-t interval* uses the quantiles of

$$\frac{\hat{\theta}^* - \hat{\theta}}{s(\hat{\theta}^*)};$$

- The *symmetric percentile-t interval* uses the quantiles of

$$\frac{|\hat{\theta}^* - \hat{\theta}|}{s(\hat{\theta}^*)};$$

The convergence of bootstrap estimates is $O(n^{-1/2})$, but the $n^{-1/2}$ term is even and the n^{-1} term odd: this can lead to cancellations in certain situations – notably, the percentile-t interval is in $O(n^{-1})$ and the the symmetric percentile-t interval is in $O(n^{-3/2})$.

To bootstrap a regression $y = x'\beta + e$, sample the noises e_i^* from the residuals \hat{e}_i and resample the predictors (or leave the predictors as-is) – but this assumes that predictors x and noise e are independent.

The *wild bootstrap* uses a 2-point distribution for each e_i^* so that the first three (conditional) moments of e_i^* coincide with those of \hat{e}_i :

$$P \left[e_i^* = \frac{1 \pm \sqrt{5}}{2\sqrt{5}} \right] = \frac{1 \mp \sqrt{5}}{2\sqrt{5}}.$$

Empirical likelihood (EL) maximizes the likelihood of the sample, seen as a multinomial model, $\sum \log p_i$, subject to $\sum p_i = 1$ and all the *estimating equations* you

have. For instance:

$$\begin{aligned} &\text{Maximize } \sum \log p_i \\ &\text{such that } \sum p_i = 1 \\ &\quad y - x'\beta = 0 \end{aligned}$$

The actual optimization requires *Lagrange multipliers* and is not straightforward. The EL estimator is asymptotically efficient and can be used to perform likelihood ratio (LR) tests.

The rest of the book is less clear and/or more superficial.

The presentation of the *generalized method of moments* (GMM) (B. Hansen invented it in the 1980s) is by far the least readable chapter.

An *endogeneous* model is a model $y = z'\beta + e$ with $E(ze) \neq 0$. For instance:

- You want to estimate β in $y = x'\beta + e$ but you only observe the predictor x with noise: $z = x + u$ (with OLS, the forecasts of y will be fine, but the estimation of β will be biased);
- There is a *feedback* in the system you are modeling:

$$\begin{aligned} q &= \beta_1 p + e_1 \\ p &= \beta_2 q + e_2; \end{aligned}$$

naive projection then suffers for the *simultaneous equations bias*: $\beta_1^* = (\beta_1 - \beta_2)/2$

One solution to the endogeneity problem is to add variables for which $E(xe) = 0$ – these are called *instrument variables* (IV). The estimation can be done by GMM or EL.

The presentation of the *IV estimator* and the *indirect least squares* (ILS) is confusing.

In case of homoskedasticity, $E(e_i^2 | x_i) = \sigma^2$, one can use the *two-stage least squares* (2SLS) estimator: regress z on x , then y on \hat{z} . The 2SLS bias increases with the number of IV and decreases with sample size.

The time series chapters are too superficial but cover AR, unit roots, cointegration and Granger causality.

Limited dependant variables (LDV), i.e., binary (logit, probit; which can be interpreted as a *latent variable model*), multinomial, integral (Poisson regression), censored are mentioned, but only estimated with parametric maximum likelihood, even though semi-parametric methods are superior.

The *Tobit model* is the censored regression

$$\begin{aligned} y_i &= (y_i^* > 0) ? y_i^* : 0 \\ y_i^* &= x_i'\beta + e_i \\ e_i &\sim N(0, \sigma^2) \end{aligned}$$

The *sample selection* problem is similar:

$$\begin{aligned} y_i &= x_i'\beta + e_i \\ T &= \mathbf{1}(z_i'\gamma + f_i) \end{aligned}$$

and y is only observed when $T = 1$. Since $E[e | T = 1] = \rho\lambda(z'\gamma)$, where $\rho = \text{Cov}(e, f)$, $\lambda = E[\varepsilon | \varepsilon > -x] = \phi(x)/\Phi(x)$, $\varepsilon \sim N(0, 1)$ (λ is called the *Mills ratio*), it is natural to estimate β and ρ using

$$y = x'\beta + \rho\lambda(z'\gamma) + u$$

$$E[u | T = 1] = 0$$

where γ is estimate with a probit model – this 2-step procedure is the *Heckit estimator* and it has a few robustness problems.

Only two *panel* data models are presented:

$$y \sim (1 | \text{subject}) + x$$

$$y \sim (1 | \text{subject}) + \text{lag}(y) + x$$

The quality of a *kernel density estimation* can be measured by the bias, the variance, their sum (asymptotic MSE or AMSE) (but all this depends on the point) or the integral of that sum (integrated AMSE or AMISE)

$$\text{Bias}(x) = \frac{1}{2}f''(x)h^2\sigma_K^2$$

$$\text{Var} \hat{f}(x) \approx \frac{f(x)R(K)}{nh}$$

$$\text{AMSE}(x) = \text{Bias}(x) + \text{Var} \hat{f}(x)$$

$$\text{AMISE}(x) = \int \text{AMSE}(x) dx$$

where K is the kernel, $R(K) = \int K^2$ its *roughness* and h the bandwidth. This can be used to choose the bandwidth.

Bootstrap methods and permutation tests **T. Hersterberg et al.**

This is part of the S-Plus documentation but remains software agnostic. It gives some good advice (use bootstrap confidence intervals corrected for bias and skewness; permutation tests are not valid if, under the null hypothesis, the two samples do not have the same spread and shape), but fails to give any technical detail (you will not know how *BCa* and *tilting* confidence intervals are computed).

Developing time-oriented database applications in SQL **R.T. Snodgras** **Morgan Kaufman (2000)**

This is an SQL cookbook for temporal problems. Its examples are complicated, anti-pedagogic (lots of numeric magic numbers, numeric codes, abbreviations and cryptic acronyms as column names, only chunks of the database at a time so as to prevent a global view of the data, examples intentionally “quite difficult to decipher”, etc.) and dusty (some references to Cobol).

There are three temporal types (instants, intervals and periods (anchored intervals)); three types of times (user-defined (a normal piece of data), valid time

(time in the real world), transaction time (time in the database)); three types of statements (current, sequenced, non-sequenced). Time in SQL is *granular* (i.e., discrete, e.g., with a 1-day or 1-minute precision).

NOW is an unsolved problem in SQL: you can either use NULL, but this has side effects for comparisons and clashes with other uses of NULL or use a magic value far in the future, which, besides its arbitrariness, leads to a confusion between “until further notice” and “forever”.

A temporal table is a table with **start** and **end** columns. A *temporal partition* splits this table into current (the **end** column contains NOW) and past tables (this is actually a solution to the NOW problem). A *snapshot table* is a slice of a temporal table, at a given date. A *transaction table* contains a log of the changes to a table (similar to a set of patches in a version control system). You might want to vacuum transaction tables or even temporal tables, if old data becomes irrelevant.

A *bitemporal table* contains both valid time and transaction time: there are two **start** and two **end** columns. A bitemporal table can be partitionned along valid time, transaction time or both.

Most of the book is devoted to the handling, in SQL, of temporal or bitemporal tables: sequenced JOIN,

```
SELECT A.x, B.y, ...
      last(A.start, B.start) AS start,
      first(A.end, B.end) AS end
FROM A, B
WHERE ...
AND last(...) < first(...);
```

sequenced EXCEPT, sequenced DISTINCT (trickier: procedural), sequenced PRIMARY KEY, sequenced UPDATE, etc. – if in doubt, draw all the possible relative positions of the intervals and write a huge query full of UNION or CASE.

Bitemporal tables complicate things, but not as much as one would fear. Bitemporal diagrams are bidimensional.

The recipe to build a temporal database goes as follows:

- Build a non-temporal ER (entity-relation) model;
- Add time;
- Convert it into a relational model (no more entities, only relations);
- Convert it to a physical model (with tables, taking into account efficiency problems, e.g., temporal partitioning).

To deal with time in an application using the database, choose among:

- Avoid time;
- Write unreadable SQL code;
- Use a middleware to convert SQL/Temporal (a rarely implemented part of the SQL standard) to SQL.

The book ends with a presentation of SQL/Temporal: it adds a PERIOD data type,

```
PERIOD '[2001-01-01 2002-01-01]';
```

with a few operators (OVERLAPS, PRECEDES, SUCCEEDS, MEETS, BEGIN, END, LAST, PRIOR, NEXT, P_UNION, P_INTERSECT, P_EXCEPT, CAST (to change the granularity)), a VALIDTIME keyword to handle temporal tables

```
ALTER TABLE T1 ADD VALIDTIME(DATE);
ALTER TABLE T1 ADD
  VALIDTIME PRIMARY KEY (x,y);
ALTER TABLE T1 ADD
  VALIDTIME UNIQUE (x,y);
ALTER TABLE T1 ADD
  VALIDTIME FOREIGN KEY (f) REFERENCES T2;
VALIDTIME SELECT ... FROM ...;
NONSEQUENCED VALIDTIME SELECT ...
  WHERE VALIDTIME(T1)
  OVERLAPS DATE '2001-01-01';
```

a TRANSACTIONTIME keyword

```
VALIDTIME AND NONSEQUENCED TRANSACTIONTIME
SELECT ...
```

EXPAND and NORMALIZE (for sets of potentially overlapping or abutting periods) operators and a NORMALIZE ON construct (similar to GROUP BY).

You may note that SQL/Temporal fails to address the NOW problem.

***The statistical bootstrap
and other resampling methods***
P. Burns (2007)

A short introduction to the bootstrap.

Dart to the heart
P. Burns (2007)

A advert for random portfolios, comparing them to benchmarks and peer groups.

***Total return strategies
for multi-asset portfolios:
dynamically managing portfolio risk***
U. Herold et al.
Journal of portfolio management (2007)

This article review various dynamic asset allocation (DAA) strategies:

- Portfolio insurance (stop loss, synthetic put, CPPI);
- Rainbow options (best-of-two, best-of- n plus floor);
- Optimization with a constraint on the value at risk (VaR) or the expected shortfall (ES, CVaR), e.g., require that the probability of going below a pre-determined floor be below 1%.

The VaR-based strategies seem preferable, provided one controls the turnover.

***A Cardan's discriminant approach
to predicting currency crashes***
S.K. Koh, W.M. Fong and F. Chan
Journal of international money and finance
(2007)

The *generalized normal* (GEN) distribution,

$$f(u) \propto \exp \left(\theta_1 u + \theta_2 \frac{u^2}{2} - \frac{u^4}{4} \right),$$

allows for bimodality and can be used to model large jumps. Bimodality can be measured with *Cardan's determinant*

$$CD = \frac{\theta_1^2}{4} - \frac{\theta_2^3}{27}$$

which can help predict currency crises – both type I and type II error rates are low.

The same idea could be generalized to any other jump prediction situation.

Towards reliable efficient frontiers
K. Schöttle and R. Werner
Journal of asset management (2006)

One could use *robust optimization*:

- For the portfolio to change slowly with time, when the inputs are changed;
- For the portfolio to change slowly when changing the risk target;
- To account for parameter estimation error.

Robust optimization outperforms *Michaud's resampled portfolio*, which outperforms classical mean-variance optimization.

***Incorporating estimation errors into portfolio
selection: robust portfolio construction***
S. Ceria and R.A. Stubbs
Journal of asset management (2006)

The list of alternatives to mean-variance portfolio optimization:

- *James–Stein estimators* shrink the expected returns of each asset to the average expected returns, depending on the volatility of the asset;
- *Jorion estimators* shrink the expected returns estimate towards the minimum variance portfolio;
- The *Black–Litterman* model blends views on some portfolios to implied market returns;
- *Michaud's resampled portfolios* are computed as follows: on bootstrap samples, estimate average returns and variance matrix, compute the optimal portfolio, average those portfolios;
- Adding constraints might improve (or worsen) the sensitivity of the portfolio to parameter changes;
- Robust optimization, where we only give the optimizer intervals containing the alpha;

the authors add a new one.

One can estimate the error on the expected returns of the portfolio and maximize this return with a penalty for the estimation error:

$$\begin{array}{ll} \text{Maximize} & \mathbf{w}'\boldsymbol{\alpha} - \lambda \left\| \Sigma^{1/2} \mathbf{w} \right\| \\ \text{st} & \mathbf{w}'V\mathbf{w} \leq v \\ & \mathbf{w}'\mathbf{1} = 1 \\ & \mathbf{w} \geq 0 \end{array}$$

where λ is the penalty for the estimation error, Σ is the variance matrix of the estimation errors, V is the variance matrix of the returns, $\boldsymbol{\alpha}$ are the forward returns, v is the risk target and \mathbf{w} are the portfolio weights the optimizer should find.

This is not a quadratic problem, but a *second-order cone program*.

***Improving investment performance
for pension plans***
J.M. Mulvey et al.

Journal of asset management (2006)

We do not live in a one-period world; traditional risk-reward measures such as the Sharpe ratio are not directly applicable to a multi-period setup; one should consider multi-period strategies, stochastic programming, etc.

***Investing in hedge funds: adding value through
active style allocation decisions***

L. Martellini et al. Edhec, SGAM (2005)

The Black–Litterman framework can be generalized to allow views on higher moments; the optimization can then use a fourth-order approximation of the utility function and the (Cornish-Fisher) VaR as a risk measure.

***An analysis of performance measures
using copulae***

S. Hwang and M. Salmon (2001)

The following performance measures

- *Jensen's alpha*, i.e., the intercept in the regression of the returns (we always consider the returns minus the risk-free rate) against the market returns (this is the alpha in the CAPM);
- The *Treynor and Mazuy measure*, the intercept and the quadratic term in a quadratic CAPM,

$$\begin{aligned} r_t &= \beta_1 r_t^{\text{market}} + \beta_2 (r_t^{\text{market}})^2 + \varepsilon_t \\ \text{TM} &= \alpha + \beta_2 E[(r_t^{\text{market}})^2] \end{aligned}$$

- The higher moment performance measure (?)

$$\begin{aligned} \text{HM} &= \mu + \lambda_1 \mu_M + \lambda_2 (\beta_{P,M} - \gamma_{P,M}) \\ \mu &= E[r_t] \\ \mu_M &= E[r_t^{\text{Market}}] \\ \sigma_M^2 &= \text{Var}(r_t^{\text{Market}}) \\ \gamma_M &= \frac{\text{Cov}(r_t^{\text{Market}}, r_t^{\text{Market}}, r_t^{\text{Market}})}{\sigma_M^4} \\ \theta_M &= \frac{\text{Cov}(r_t^{\text{Market}}, r_t^{\text{Market}}, r_t^{\text{Market}}, r_t^{\text{Market}})}{\sigma_M^4} \\ \beta_{P,M} &= \frac{\text{Cov}(r_t, r_t^{\text{Market}})}{\sigma_M^2} \\ \gamma_{P,M} &= \frac{\text{Cov}(r_t, r_t^{\text{Market}}, r_t^{\text{Market}})}{\sigma_M^3} \\ \lambda_1 &= \frac{\gamma_M^2 \gamma_{P,M} - \theta_M}{\gamma_M^2 - (\theta_M - 1)} \\ \lambda_2 &= \frac{\gamma_M \sigma_M}{\gamma_M^2 - (\theta_M - 1)} \end{aligned}$$

- The positive weighted weighting (PPW) measure (?)

The Sharpe ratio is very different from the others and, worryingly, for extreme positive values, those performance measures are *asymptotically independent*.

To this end, the article reviews various measures of tail or asymptotic dependence, around

$$\lambda = \lim_{u \rightarrow 1} P[Y > F_Y^{-1}(u) | X > F_X^{-1}(u)].$$

To obtain a more reliable measure of performance, one can use a *copula*, describing the relation between those measures, to combine them, yielding the following distribution (the choice of the copula is left to the reader, e.g., a gaussian copula to simplify the computations):

$$f(x|f1, f2) = f_1(x)f_2(x)c(1 - F_1(x), 1 - F_2(x)).$$

***Stock returns and volatility:
pricing the short-run and long-run
components of market risk***

T. Adrian and J. Rosenberg (2006)

Volatility can be modeled by a GARCH-like model where $\log v_t$ is the sum of two AR(1) processes.

***Market reactions to tangible
and intangible information***

K. Daniel and S. Titman
Journal of finance (2006)

The *intangible return*, i.e., the component of the past returns unexplained by the firm's past performance, defined as the residuals of the regression (over 5 years; the book-value growth is actually corrected for dividend payments)

$$\text{forward returns} \sim \log(B/P) + \text{growth}(B)$$

is negatively related to future returns.

The article also explains why the B/P helps predict future returns: it can be written as

$$\log(P/B)_{\text{now}} = \log(B/P)_{\text{before}} + \text{growth}(B) - \text{growth}(P)$$

(where $\text{growth}(x) = \log(x_{\text{now}}/x_{\text{before}})$) and only $\text{growth}(B) - \text{growth}(P)$ has a predictive power.

Predicting the time-varying covariance matrix of electricity forwards

K. Aas and K.F. K arsen (2003)

The ARCH model can be generalized to describe the covariance between similar, short-lived, overlapping time series (forwards).

Trimability and fast optimization of long-short portfolios

B.I. Jacobs et al.

Financial analysts journal (2006)

The authors explain how to transform most long-short optimization problems into a long-only problem, suitable for long-only optimizers.

Feedback and the success of irrational investors

D. Hirshleifer et al.

Journal of financial economics (2006)

It is good to be an early irrational investor and bad to be a late one – provided many share your irrationality. Early irrational investors have private information about the future order flow of investors with similar biases. This effect causes prices to follow a random walk...

A comparison of financial duration models via density forecasts

L. Bauwens et al. (2000)

Duration data are irregular time series represented as bidimensional time series, $(x_n, t_n - t_{n-1})$, where x_n is the value measured at time t_n . Examples include price or volume in high-frequency financial data. They exhibit properties similar to those of GARCH data, such as *duration clustering*: hence, one can model them along the same lines by replacing the (unobserved) volatility by the (observed) duration – GARCH becomes ACD (autoregressive conditional duration), T-FARCH becomes TACD (threshold ACD), log-GARCH becomes log-ACD and stochastic volatility models become SVD (stochastic volatility duration).

The authors compare those methods using *density forecasts*: prediction methods do not simply provide a forecast value \hat{x}_i , but a whole forecast distribution \mathcal{F}_i : for each observation i , you can compute $p_{\mathcal{F}_i}(x_i)$, where x_i is the actual observation and $p_{\mathcal{F}_i}$ is the p -function (cumulative distribution function) of the predictive distribution. If the forecasts are correct, the $p_{\mathcal{F}_i}(x_i)$ should be uniformly distributed – this can be tested. This method accounts for models that change over time.

Equity style timing using support vector regressions

G. Nalbantov et al.

Support vector machines (SVM) can be used to *time* growth vs value and large vs small strategies.

The variables used are: value-growth spread, small-large spread, S&P 500 volatility, 12-month forward S&P 500 P/E, 6-month S&P 500 momentum, EPS change over the past year in the S&P 500, P/E differences between value and growth, P/E differences between large and small caps, dividend yield difference between value and growth, dividend yield difference between large and small caps, yield spread Baa vs Aaa, 12-month change in the US CPI (consumer price index), difference between the forward E/P of the S&P 500 and the 10-year T-bond yield, yield spread between 10-year T-bonds and 3-month T-bills, 10-year T-bond yield adjusted for the inflation rate, seasonally-adjusted US industrial production, and a couple more.

Applying financial statement analysis to forecast earnings growth and evaluate P/E ratios

S. Li (2003)

If investors buy future earnings, the P/E should be explained by the earnings (or operating income) growth, which can be obtained by a big cross-sectional regression with the variables from the financial statement. The author suggests to combine this estimate with that from the analysts.

Essential portfolio theory

J. Mulvey

Rydex investments (2005)

Empty article: it just reminds us that we do not live in a long-only, equity-and-bonds-only, 1-period, static, unleveraged world.

You want to diversify risk? Consider economic derivatives

R. Dubil

Journal of wealth management (2007)

Individual investors should use *economic derivatives* (consumer inflation, currencies, real estate prices, gas-at-the-pump, etc.) to hedge their non-stock assets (house, purchasing power, etc.).

Contrary to derivatives with a tradable underlying, economic derivatives are valued with *probabilistic insurance pricing*.

An introduction to differential geometry in econometrics

P. Marriott and M. Salmon

in *Applications of differential geometry to econometrics*, Chapter 1 (CUP, 2000)

A parametric (exponential) family of probability distributions can be seen as a manifold (generalizations

to the manifold of “all” distributions are possible, but infinite-dimensional manifolds pose further theoretical problems) and statistical properties of distributions, tests or estimators can be translated into geometric properties, for instance:

- The *score* statistic is the length of a certain cotangent vector measured with respect to the metric defined by the expected Fisher information;
- The Kullback–Leibler divergence defines a (non-metric) *connection*;
- The *curvature* of the manifold is linked to information loss (on a flat statistical manifold, expected and observed information coincide).

The authors do not mention the interpretation of *robust* estimators as continuous functions on a statistical manifold.

***Region vs industry effects
and volatility transmission***

**P. Soriano and F. Climent
Financial analysts journal (2006)**

This article studies *volatility transmission* (anova) between regions, between industries, between regions within industries, between industries within regions, and confirms the dominance of region effects over industry effects.

***Common failings:
how corporate defaults are correlated***
S.R. Das et al. Journal of finance (2007)

Defaults cluster in time, but this cannot be explained only by the exposure to common risk factors (the authors develop a test for this): *contagion* plays a role.

***Cluster analysis:
two macro themes dominate Eurozone markets***
**A. Harmstone
Bear Stearns (2005)**

One can cluster stocks (with the PAM algorithm) using their exposure to risk factors and then use the resulting (two) groups as *macro themes* (depending on the market conditions, these could be value and growth, or something entirely different).

***Information transmission
across global markets***
**P. Singh
Bear Stearns (2006)**

Uncertainty (volatility) is transmitted both ways between Asia (Japan, Hong Kong, Taiwan, South Korea) and the West (US and Europe), while returns are mainly transmitted from Asia to the West with a surprising 1-day lag. As a result, intra-day US strategies should use Asian information.

***Implied macro returns:
profiting from over-reaction***
**A. Harmstone
Bear Stearns (2006)**

Yet another article advocating the use of cumulated (time series) residuals.

***Multilayer modeling of
a market covariance matrix***
**R. Staub
Journal of Portfolio management (2006)**

Rolling (or weighted rolling) volatility is a bad estimate of future volatility: prefer GARCH modeling.

One can generalize strict factor models (“risk models”) in a multilayer way: estimate the covariance between asset groups; then, within each group, between the subgroups; etc..

***Liquidity and stock returns:
evidence from a pure order-driven market
using a new liquidity proxy***
**B.R. Marshall
International review of financial analysis
(2006)**

The *weighted order value* (WOV) is a new liquidity proxy: in 30-minute windows, bin the bids (and asks) with respect to the best one (e.g., 1% from the best, 2%, 5%, 10%, etc.); compute the bid (resp. ask) execution rate (number of executed orders divided by the number of orders) in each bin; pool the bids in each bin to get the bid order value (the sum of the bid prices times the bid volumes); pool the bins: the weighted bid value is the sum of the products of the bid order values and the bid execution rates; the WOVI is then the geometric mean of the bid and ask weighted values.

Sharpening Sharpe Ratios
W. Goetzmann et al. (2004)

Sharpe ratios are amenable to manipulations, with options (non-linear payoffs), dynamic strategies (that also modify the distribution of returns: truncate it on the right and fatten the left tail to the mean to the right) or smoothed returns.

***Futures trading volume as a determinant of
prices in different momentum phases***
**A. Hodgson et al.
International review of financial analysis
(2006)**

Options trading volume has predictive power on the short-term (15 minutes) stock price; this effect changes with the current price trend.

***Advanced frequency and time domain filters
for currency portfolio management***
**C. Dunis and J. Miao
Journal of asset management (2006)**

Many FX traders use a combination of several MACDs (say, 1 and 32 days, 1 and 61 days, 1 and 117 days, equal-weighted), but this strategy does not perform well.

Spectral analysis (the periodogram) can help identify cyclical spells in FX time series (you can try to use it as a trading strategy, but it will not work).

The article suggests to follow the MACD strategy but to modify it in cyclical and/or volatile (as measured by the RiskMetrics model, *i.e.*, an exponentially-weighted non-central variance) periods, by not trading or trading in the opposite direction.

***On the maximum drawdown
of a brownian motion***
M. Magdon-Ismail et al. (2003)

The (asymptotic) expected maximum drawdown (MDD) of a brownian motion is

$$\begin{aligned} X(t) &= \sigma W(t) + \mu t \\ E[D] &= \sqrt{\frac{\pi}{2}} \sigma \sqrt{T} && \text{if } \mu = 0 \\ E[D] &\sim_{T \rightarrow \infty} \frac{\sigma^2}{2\mu} \log T && \text{if } \mu > 0 \\ E[D] &\sim_{T \rightarrow \infty} -\mu T && \text{if } \mu < 0. \end{aligned}$$

Maximum drawdown
M. Magdon-Ismail et al. (2003)
Risk (October 2004)

Using the previous article, one can study the term structure of the *Calmar* or *Sterling* ratios.

$$\begin{aligned} \text{Calmar} &= \frac{\text{returns on } [0, T]}{\text{MDD on } [0, T]} \\ \text{Sterling} &= \frac{\text{returns on } [0, T]}{\text{MDD on } [0, T] - 10\%} \end{aligned}$$

The misuse of expected returns
E. Hughson et al.
Financial analysts journal (2006)

Yet another article against ratio returns...

At the very least, if you have to deal with ratio returns, prefer their median to their mean.

***Portfolio selection with parameter and model
uncertainty: a multi-prior approach***
L. Garlappi et al.
Review of financial studies (2007)

The article interprets the robust portfolio optimization framework

$$\begin{aligned} &\text{Maximize} && \min_{\alpha} w' \alpha - \lambda w' V w \\ &\text{st} && \alpha_{\min} \leq \alpha \leq \alpha_{\max} \end{aligned}$$

in a bayesian setting, as a generalization of the Black–Litterman model, in terms of *uncertainty aversion*.

***Common and country-specific components
in national stock prices***
O. Hu
Journal of multinational financial management
(2006)

Market integration can be measured by cointegration – more precisely, the presence of a *single* cointegration relation.

G7 countries are not integrated: diversification benefits still exist.

***Testing for multiple regimes in the tail
behavior of emerging currency returns***
B. Candelon and S. Straetmans
Journal of international money and finance
(2006)

The *tail index* of a distribution is the parameter α such that

$$\forall x > 0 \quad \lim_{t \rightarrow \infty} \frac{1 - F(tx)}{1 - F(t)} = x^{-\alpha},$$

i.e., such that the tail of the distribution be a first order approximation of a Pareto distribution ($F_{\text{Pareto}}(x) = 1 - ax^{-\alpha}$).

The *Hills estimator* of the tail index in a sample of size n is

$$\hat{\alpha} = \left(\frac{1}{m} \sum_{j=0}^{m-1} \log \frac{X_{(n-j)}}{X_{(n-m)}} \right)^{-1}$$

where the $X_{(k)}$ are the order statistics and m is chosen so that m/n be small (the estimator has good asymptotic properties for $m/n \rightarrow 0$ and $n \rightarrow \infty$).

The authors then perform *recursive tests for structural breaks*: when you find a break, split the sample in two and look for new breaks in each chunk.

There are structural breaks in the tail index of emerging market currency returns.

***Are international value premiums driven by
the same set of fundamentals?***
A.J. Black et al.
International review of economics and finance
(2007)

Yet another article that gauges the benefits of diversification (of a value strategy, across G7 countries) using cointegration: cointegration is present.

***Using dynamic programming
to optimally rebalance portfolios***
W. Sun et al.
Journal of trading (2006)

Dynamic trading strategies include: periodic rebalancing, *tolerance band rebalancing*, *no trade zone* (identical to the tolerance band rebalancing, but the band is not arbitrarily defined and we do not rebalance all the way to the optimal portfolio but just back into the band).

The authors measure, in USD, the tracking error of a portfolio as the risk-free return giving the same utility (*certainty equivalent*) and use this measure in a dynamic program to control rebalancing – this can be seen as a dynamic equivalent of tolerance band rebalancing. The problem is amenable up to 8 or 9 assets.

***Forecasting online auctions
using dynamic models***

W. Jank et al.

**KDD 2006 Workshop on Data mining
for Business Applications**

The authors use *functional data analysis* (FDA) to forecast irregular time series (ebay auction prices) taking into account the influence of opening bid, auction length and seller's reputation on the price dynamics (acceleration, velocity and eventually final price).

***Data mining in the real world:
what do we need and what do we have?***

F. Fogelman Soulié

**KDD 2006 Workshop on Data mining
for Business Applications**

Data mining is widely used for CRM (client relation management), fraud detection, credit scoring, etc. and *extreme data mining*, *i.e.*, data mining with databases beyond 10 TB is starting to emerge. It is characterized by:

- Heterogeneous data;
- poor data quality (outliers, missing data and even *unlabelled data*, *i.e.*, “massively missing values”);
- Large volumes (millions of rows, thousands of variables) that prevent the user from hand-chosing the variables and calls for *data duplication avoidance*;
- Fast model calibration (hours; linear in the number of rows or columns);
- Automated model quality assessment (they use a *robustness indicator*, KR , from Vapnik's *structural risk minimization theory*);
- Real-time model application;
- Industrialization (“productionalization”) within the same project, not as a separate, long project;
- Model control (checks deviations in the data distribution);
- Data export using data mining standards (JDM, PMML);
- Process control and workflow;
- Exploratory modeling (the end-user needs to understand the model);
- Predictive modeling (the end-user want accurate forecasts);
- Unskilled users.

The article failed to mention the following:

- Asynchronicity (we might want real-time results even though some data sources might not be real-time; bursts of data may delay the delivery of the results);
- Distributed analysis (the volume and location of the

data, with the requirement that the data should not be duplicated, will call for the computations to be run where the data is, possibly in different locations).

***The landscape of parallel computing research:
a view from Berkeley***
2006

The next big thing in computer science could be *distributed computing*: this article enumerates benchmarks (called “*dwarfs*”) or challenges for such systems.

Among the potential “next big thing”, one could add, in no particular order: declarative programming, functional languages, virtualization, virtual machines.

DNA computers, quantum computers are still decades (or centuries) away, though programming paradigms are already being developed for those architectures.

Concurrency and Erlang
A. Pang

Another article about distributed programming and the need for adequate tools – not threads: in the hands of most programmers, they make $i = i + 1$ non-deterministic...

On the same subject:

- Some real-time systems already use lock-free programming, *e.g.*, the Jack audio server, as explained in the Linux Audio Conference 2005.
- Software Transactional Memory (STM) is getting more widely discussed, and is already in the Haskell libraries.

***Robust moving least squares fitting
with sharp features***
S. Fleishman et al.
SIGGRAPH 2005

This article applies an *outlier detection* algorithm (the *forward search algorithm paradigm*: start with a small set of robustly chosen points, progressively add more points, while monitoring some statistic) for *surface reconstruction*, accounting for noise and singularities (*i.e.*, sharp edges).

***Everything you know about
dynamic time warping is wrong***
C.A. Ratanamahatana and E. Keogh
SDM 2005

This article disspsells a few DTW myths:

- DTW's ability to deal with sequences of different lengths is not an advantage (but not a disadvantage either): you would get the same results after resampling the sequences;
- The 10% constraint on warping paths inherited from the speech processing community is not too low but actually much too high for most data mining applications;
- As a result, DTW is essentially $O(n)$.

Portfolio selection in stochastic environments

J. Liu

Review of financial studies (2007)

The *dynamic portfolio choice* problem can be expressed as a partial differential equation (Merton, 1971), which does not lead to a general, amenable solution. This article derives an explicit solution in the case of *quadratic returns*, generalizing similar results for *Ornstein–Uhlenbeck processes*.

Emergent effective collusion in an economy of perfectly rational competitors

R.K. Standish and S. Keen

arXiv:nlin.AO/0411006

Some agent-based models predict the emergence of monopolies; irrational agents or delays can help avoid them.

Portfolio optimization with drawdown constraints

A. Chekhlov et al. (2003)

In portfolio optimization, one can replace the standard deviation by other risk measures such as the expected shortfall (ES, conditional value at risk, CVaR) or the conditional drawdown at risk (CDaR).

Investor sentiment and the cross-section of stock returns

M. Baker and J. Wurgler

Journal of finance (2006)

You might want to completely reverse your trading strategy (usually: large, mature, low-volatility, profitable, dividend-paying, non-distressed, value stocks) when the investor sentiment is low.

The *investor sentiment* can be computed from the following (annual) proxies (do not forget to remove the systematic risk):

- Opposite of the closed-end fund discount (CEFD), *i.e.*, the average difference between the net asset value (NAV) of closed-end stock funds shares and their market prices;
- Detrended NYSE share turnover (or any measure of liquidity);
- Number of IPOs;
- Average first-day returns on IPOs;
- Equity share in new issues;
- Dividend premium, *i.e.*, difference of the P/B of dividend payers and non-payers.

Re-examining the profitability of technical analysis with White’s reality check and Hansen’s SPA test

P.H. Hsu and C.M. Kuan (2005)

To avoid data snooping, one can:

- Claim it is not a problem (but it is!);
- Split the data into in- and out-of-sample (but this is arbitrary and discards part of the data);

- Explicitly test all the models and correct the p -values accordingly (but when there are too many tests, the power of the corrected tests drops);
- Use the bootstrap to compute the p -value:
 - Build a universe of all trading strategies you might be interested in;
 - Choose a performance measure (White uses the annualized returns, Hansen uses the information ratio or 0 when it is negative);
 - Estimate the distribution of the maximum performance on bootstrap samples;
 - Compare the performance of your model with the distribution of the best performing model.

The article applies this methodology to technical analysis (TA): it works, and works better in less mature markets.

Data Envelopment Analysis

G.N. Gregoriou and J. Zhu

Journal of portfolio management (2007)

The notion of *efficient frontier* can be generalized to higher dimensions: several risk measures or inputs (standard deviation, downside deviation, maximum drawdown), several outputs (returns, proportion of profitable months, maximum consecutive gain).

If you have enough data points, they provide an approximation of the efficient frontier (?).

The *efficiency* (closeness to the efficient frontier) can be defined as the solution of a linear problem:

Minimize θ such that

$$\sum_k \lambda_k x_{ik} \leq \theta x_{ik_0}$$

$$\sum_k \lambda_k y_{jk} \geq y_{jk_0}$$

$$\sum_k \lambda_k = 1$$

$$\lambda_k \geq 0$$

i : input

j : output

k : funds

k_0 : fund whose efficiency is being computed

x_{ik} : input i of fund k

y_{jk} : output j of fund k .

See <http://people.brunel.ac.uk/~mastjjb/jeb/or/dea.html> for a picture.

Optimization of the largest US mutual funds using data envelopment analysis

G.N. Gregoriou

Journal of asset management (2006)

Another article on data envelopment analysis (DEA, sometimes also called *frontier analysis*). It defines the

classical efficiency as

$$CCR_{k_0} = \text{Max} \left\{ \frac{\sum_j \mu_j y_{jk_0}}{\sum_i \lambda_i x_{ik_0}} \text{ st } \forall k \frac{\sum_j \mu_j y_{jk}}{\sum_i \lambda_i x_{ik}} \leq 1 \right\}$$

and the *super-efficiency* (which is no longer bounded by 1) as

$$CCR_{k_0} = \text{Max} \left\{ \frac{\sum_j \mu_j y_{jk_0}}{\sum_i \lambda_i x_{ik_0}} \text{ st } \forall k \neq k_0 \frac{\sum_j \mu_j y_{jk}}{\sum_i \lambda_i x_{ik}} \leq 1 \right\}$$

They also mention the *cross-efficiency model*, but not clearly.

Asset allocation for robust portfolios

T. Farrelly

Journal of investing (2006)

This article is a variation on Michaud's *resampled portfolios*:

- Start with your risk model and your alpha; choose a risk threshold and a “sensitivity to underperformance” λ ;
- Compute $N = 500$ simulated returns from $N(0, V)$ (or $N(\alpha, V)$, the article is not clear) and let

r_{ij} = return of stock i in situation j
 w_{ij} = weight of stock i in
the efficient portfolio ($r_{\bullet, j}, V$)

(the article is not clear here either: there sometimes seems to be one optimal portfolio per scenario, sometimes one for the set of all scenarios)

- Minimize

$$\sum_{ij} |(w_{ij} - w_i) r_{ij}|^\lambda$$

under the condition that the risk be below the threshold.

Automatic bayesian model averaging for linear regression and applications in bayesian curve fitting

F. Liang et al.

Statistica Sinica 11 (2001) 1005–1029

The article suggests an “automatic prior” for bayesian model average (BMA) – in situations where it has an edge on model selection, BMA is sensitive to the prior, for instance, to the expected number of factors in the model.

The inclusion probability still has to be provided by the user; the authors only provide a prior for β and σ (equations 10 and 11).

Evolutionary Monte Carlo: applications to C_p model sampling and change point problem

F. Liang and W.H. Wong

Statistica Sinica 10 (2000) 317–342

Check *Monte Carlo Strategies in Scientific Computing*, J.S. Liu, Springer Verlag, 2002, for a 1-page summary of this article.

Are alternatives the next bubble?

J. Loeys and N. Panigirtzoglou

Journal of alternative investment (2006)

One can use high returns, expensive valuation (low risk premium), speculative activity (trading volume, credit, leverage) as a bubble indicator.

Profitability of price momentum strategies: surprising evidence from Pacific-Bassin countries

P. Ryan and R. Curtin

Journal of investing (2006)

Momentum does not work in Asia – but have a look at reversal.

Do asset prices reflect fundamentals?

Freshly squeezed evidence from the orange juice market

J. Boudoukh et al.

Journal of financial economics (2006)

Example of the misuse of linear regression to study a non-linear phenomenon.

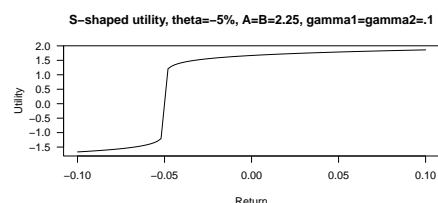
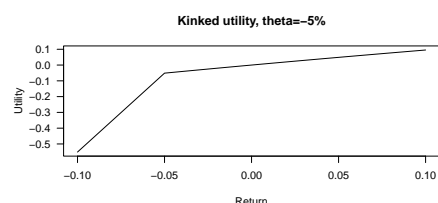
Mean-variance versus full-scale optimisation: in- and out-of-sample

T. Adler and M. Kritzman

Journal of asset management (2007)

The authors suggest to replace quadratic utility (or even power utility: $U(x) = x^\gamma/\gamma$, which translates *constant relative risk aversion* (CRRA)) by *kinked utility* (for investors wishing to breach a threshold, e.g., the minimum level of wealth to maintain a certain standard of living) or *S-shaped utility* (to reflect asymmetric risk behaviour).

$$\begin{aligned} U_{\text{kinked}}(x) &= \ln(1+x) & \text{if } x \geq \theta \\ &= 10(x-\theta) + \ln(1+\theta) & \text{if } x < \theta \\ U_{\text{S-shaped}}(x) &= -A(\theta-x)^{\gamma_1} & \text{if } x \leq \theta \\ &= -B(x-\theta)^{\gamma_2} & \text{if } x > \theta \end{aligned}$$



***Forecasting financial time series
using artificial market models***

N. Gupta et al.
arXiv:physics/0506134

Stock market simulations by *agent-based models* suggest there are *pockets of predictability*; once one such pocket is closed, investment strategies trained on previous data become worthless.

In other words: agent-based market simulations confirm the predictability of stock markets and the presence of structural breaks.

***Are you about to handcuff
your information ratio?***

R. Staub
Journal of Asset Management (2007)

If you want yet another explanation/application of the FLAM (fundamental law of active management)...

***Alphas with attitude:
revving up your factor model***

B. Lau and G. Platt
Macquarie Research Equities (2007)

Yet another article promoting dynamic, IC-based weights: $w = (\text{Var IC})^{-1} \cdot \text{IC}$, using an expanding window and imposing constraints on the weights ($0 \leq w \leq 0.5$).

Do industries lead stock markets?

H. Hong et al.
Journal of Financial Economics (2007)

One can predict market returns from lagged industry returns; this is due to the *bounded rationality* of investors: they cannot process all the information about all the stocks immediately when they receive it; the effect is present in all developed markets; the predicting power is comparable to that of inflation, default spread and dividend yield.

Others had already noticed that large or liquid stocks lead small or illiquid ones.

***Noise trader risk:
evidence from the Siamese twins***

J.T. Scruggs
Journal of Financial Markets (2007)

Identical stocks on different exchanges do not move in parallel: up to half the weekly volatility is due to noise trading.

Thus, arbitrage is limited and the risk of pairs trading is high.

***Behavioral finance: frame-dependant
decision making and factor anomalies***

T. Suwabe et al.
Goldman Sachs (2006)

Investors look at their *unrealized gains* when deciding to leave a position: this can be exploited by comparing the current price of a stock with its average buying price

$$R_{t+1} = V_t P_t + (1 - V_t) R_t$$

where P_t is the price and V_t the turnover.

This is actually a momentum computed with an exponential moving average of the price (instead of the lagged price), where the coefficient depends on the volume.

Profiting from momentum

R. de Souza
Macquarie Research, Equities (2006)

Another article on the same subject, under the name *capital gains overhang*.

***Alpha repair: a factor competition
approach to stock selection***

J.J. Mezrich and J. Feng
Nomura (2007)

To select the factors to put in your model, regard them as assets, evaluate their long-term (5-year) returns, their variance matrix, and feed them to your preferred optimizer to find an optimal “portfolio of 5 factors” (you can fine-tune the number of factors).

***Refinements to the Sharpe ratio:
comparing alternatives for bear markets***

H. Scholz
Journal of Asset Management (2007)

Sharpe ratios (excess returns divided by risk) cannot be used to compare funds when the returns are negative (they rarely give the right order).

The *normalized Sharpe ratio* tries to correct this. The Sharpe ratio can be computed as

$$\begin{aligned} \text{er}_t &= \text{excess returns} \\ \text{mer}_t &= \text{market excess returns} \\ \text{er}_t &= \alpha + \beta \text{mer}_t + \varepsilon_t \\ \text{SR} &= \frac{\alpha + \beta \overline{\text{mer}}}{\sqrt{\beta^2 + \sigma(\text{mer})^2 + \sigma(\varepsilon)^2}} \end{aligned}$$

The authors simply use 20 years for $\overline{\text{mer}}$ and $\sigma(\text{mer})$.

***Momentum, reversal
and the trading behaviors of institutions***

R.C. Gutierrez and C.A. Prinsky
Journal of Financial markets (2007)

Cumulated residual returns (for a 5-year CAPM model) can help devise a long-term strategy, while the other component of the momentum (“relative momentum”) reverts after one year.

More precisely, the article builds a strategy from the signal :

$$\alpha = \frac{\text{cumulated residual returns}}{\text{cumulated variance of the residuals}}$$

This effect is linked to institutional ownership.

Countries versus industries in emerging markets: a normative portfolio approach
J. Estrada et al., Journal of investing (2006)

The authors compare dispersion, information ratio ($IR = \mu/\sigma$), utility ($U = \log(1+\mu) - \frac{1}{2}\sigma^2/(1+\mu)^2$), option price (an option that allows its owner to swap the performance of the top (or bottom) quartile with the median performance) of random country portfolios and random industry portfolios and conclude that country effects prevail.

On the use of hypothesis testing and other problems in financial research
F. Gómez-Bezares et al.
Journal of investing (2006)

The authors notices the following problems:

- Not enough “case studies” – *i.e.*, you should look at the data;
- Unnecessary mathematical formulation;
- Rarely applicable results: when they become familiar with a method (say, regression), researchers try to apply it everywhere;
- Bad null hypothesis: it may even overlap with the alternative hypothesis – the use of *confidence intervals* besides tests would mitigate the problem.
- Data snooping: be extremely suspicious of “tables of *p*-values”;
- Publication bias, *i.e.*, unpublished “tables of *p*-values”;
- Errors in the data – you should look at the data and use *robust methods*;
- Simplifying hypotheses (no transaction costs, no bid-ask spread, etc.);
- Insufficient data (sample too small);
- Non-parametric tests – only use them when needed: their power is much lower;
- Some studies turn qualitative variables into quantitative ones in a rather arbitrary way.

The authors fail to notice the following problems:

- Unnecessary data binning;
- Use of linear or gaussian methods without checking if it makes sense.

Theory and techniques of Electronic Music
M. Puckette (2007)

A hands-on digital signal processing (DSP) book, covering, among other things, filters and Fourier analysis.

Divergence of opinion and equity returns under different states of earnings expectations
J.A. Doukas et al.
Journal of financial markets (2006)

The interaction between analysts estimates and analysts divergence is non-linear.

Was there a Nasdaq bubble in the late 1990s?

L. Pástor and P. Veronesi
Journal of financial economics (2006)

The Nasdaq bubble can be explained by uncertainty about future profitability which, by Jensen’s formula, increases a firm’s value:

$$\frac{\text{Price}}{\text{Dividends}} = \frac{1}{\text{discount rate} - \text{growth}}$$

$$E[P/D] > \frac{1}{\text{discount rate} - E[\text{growth}]}$$

Analysts’ selective coverage and subsequent performance of newly public firms
S. Das et al.
The journal of finance (2006)

Initial public offerings (IPO) followed by many analysts outperform those followed by fewer analysts over three years.

Earnings and price momentum
T. Chordia and L. Shivakumar
Journal of financial economics (2006)

The momentum effect can be explained by *earnings surprises*.

Decomposing the price-earnings ratio
K. Anderson and C. Brooks
Journal of asset management (2006)

The authors suggest to correct the P/E for sector and size.

Conditional skewness in asset pricing tests
C.R. Harvey and A. Siddique
Journal of finance (2000)

Systematic skewness commands a risk premium, on average 3.6% per year; the momentum effect is related to systematic skewness.

On the aggregation of local risk models for global risk management
G. Anderson et al. (2005)

To build an integrated risk model, *i.e.*, a risk model covering several markets, from a model for each market and a global model, with the drill-down property, *i.e.*, such that the restriction of the integrated model be the local ones, one can write the global risk model as $V = BB'$ and replace B by LB where L is to be determined.

The possible values are $L = \bigoplus_i \tilde{\Theta}_i^{1/2} O_i \tilde{\Theta}_i^{-1/2}$ with $O_i \in O(n_i)$, the Θ_i are the diagonal blocks of the global model and $\tilde{\Theta}_i$ are the desired diagonal blocs – the local risk models.

Choose the O_i to minimize $\|BB' - LBB'L'\|^2$ – or rather, the distance between the corresponding correlation measures.

This is the *double Procrustes problem*.

***On pricing derivatives in the presence
of auxiliary state variables***
J. Lin and P. Ritchken
Journal of derivatives (2006)

The pricing of certain options requires a path-dependant auxiliary variable (e.g., the volatility when the underlying follows a GARCH model). This can be implemented on a lattice (often improperly called a “binomial (or trinomial) tree”) by storing an array of values at each node. The article explains that keeping a single value, viz. the expectation of the auxiliary variable, is actually sufficient, and details the Heath-Jarrow-Merton (HJM) model.

***Using order statistics to estimate confidence
intervals for probabilistic risk measures***
K. Dowd, Journal of derivatives (2006)

The α -VaR is the α th quantile: it is an order statistic. If F is the cumulative distribution function of your random variable, the r th order statistic $X_{(r)}$ of a sample of size n has the cumulative distribution function

$$G_{r,n}(x) = \sum_{i=r}^n \binom{n}{i} F(x)^i (1 - F(x))^{n-i}.$$

From there, just estimate F from the data you have (parametrically or not) and estimate the confidence interval $[x_1, x_2]$ by solving $G_{r,n}(x_1) = 0.05$ and $G_{r,n}(x_2) = 0.95$.

The author claims that the same approach applies to the expected shortfall.

***The information content
of the FOMC minutes***
E. Boukus and J.V. Rosenberg (2006)

Latent semantic analysis (LSA) goes as follows:

- Convert the texts to *bags of words*, i.e., remove the *stop words*, *stem* the words, forget their order but not their number;
- Form the *term-document matrix* X : one row per term, one column per document, the term frequency x_{ij} is the number of occurrences of term i in document j divided by the number of terms in document j ;
- Compute the singular value decomposition (SVD) of X : $X = USV'$, the columns of U are “virtual documents” that can be interpreted as the *themes* of the corpus.

There are variants for the construction of the term-document matrix:

- One can forget the number of occurrences;
- One can use *global weights*, such as *TFIDF* (term frequency inverse document frequency), which replaces the term frequency x_{ij} by

$$x_{ij} \cdot \log \frac{1}{y_i}$$

where y_i is the document frequency, i.e., the number of documents containing term i divided by the number of documents.

***Order book characteristics
and the volume-volatility relation:
empirical evidence from a limit-order market***
R. Næs and J.A. Skjeltorp
Journal of financial markets (2006)

The *order book slope* influences volume, volatility and their interactions; it is a proxy for *disagreement among investors*.

Subordinated binomial option pricing
C.W. Chang et al.
Journal of financial research (2006)

Binomial trees can easily be generalised to account for stochastic volatility, by describing the price evolution with successive binomial trials up/down and trade/non-trade (this is a *trinomial tree*, the spot price S becoming uS , S or dS with probabilities gh , $1 - g$ and $g(1 - h)$), corresponding to a time change from calendar time to operational time.

***Individual equity return data from
Thomson DataStream: handle with care!***
O.S. Ince and R.B. Porter
Journal of financial research (2006)

Beware of your data source (CRSP vs TDS): clean before use.

***Momentum: does the database
make a difference?***
B. Chakrabarty and C. Trzcinka
Journal of financial research (2006)

Statistical and real profits might or might not coincide, depending on your data source (CRSP or TAQ).

Downside risk
A. Ang et al. Review of Financial Studies (2006)

Stocks with a high sensitivity to downside market movement (β_-) have higher returns.

This effect does not (entirely) come from stock with high coskewness risk having higher returns.

Downside beta can be used as an alpha, but not for high-volatility stocks.

***Transmission of information
across international equity markets***
J. Wongswan
Review of Financial Studies (2006)

Macroeconomic announcements in the US or Japan have short-lived (30 to 60 minutes) effects on volatility and volume (not returns) in Korea and Thailand.

Lehman suite of PCA risk models
J. Ruiz-Mata (2006)

They provide statistical risk models, for Europe, US, developed Asia and emerging markets, built as follows:

- Using daily (or weekly, monthly) *excess returns* over 40 (long-only) to 60 (long-short) days, they perform a factor analysis with 4 to 8 factors;
- The number of factors is determined by *random matrix theory* (RMT), *i.e.*, by comparing the distribution of the eigenvalues with that of the variance matrix of iid gaussian variables (they do not seem to realize that RMT uses principal component analysis (PCA), not factor analysis, *i.e.*, it assumes there is no stock-specific risk: this might bias the methodology towards an unduly high number of factors – they are lucky to have so few); they fail to note that the number of factors to retain also depends on the size of the universe: 4 or 5 factors might be necessary if you have several thousand US stocks, but with just 100, one might be sufficient;
- The training period is determined by computing the distance between the factor model variance matrix and the *forward* sample variance matrix (they do not use the distance between those matrices, by the difference between the forecasted risk and the realized risk, for a large number of long-short portfolios – this is very similar to the definition of a *matrix norm*, but we do not need to have time series of forward returns, one forward return for each stock suffices).

Is fundamental analysis effective for growth stocks?
P.S. Mohanram (2003)

This article lists financial variables likely to help predict future returns for growth stocks. There are classical fundamental variables:

- ROA
- ROA growth
- Cash flow ROA
- Cash flow – Net income
- Asset turnover (sales / beginning-of-period assets) growth
- Operating margin (gross margin / sales) growth
- Leverage reduction
- Current ratio (?) growth
- Equity issuance in previous year

and growth-specific variables (all sectorwise):

- Earnings stability: 5-year ROA variance
- Growth stability: 5-year sales growth variance
- Earnings growth – Sales growth
- R&D / Beginning-of-period assets
- Capital expenditures / Assets
- Advertising / Assets

Price-to-earnings ratio and expected earnings growth rate in global equity markets

G. Bakshi and A. Chan (2000)

Expected earnings growth rate have a predictive effect on P/E variations.

Rewriting history
A. Ljungqvist et al. (2006)

Some estimates in the IBES database are anonymized (when the analysts ask for it). You would expect that: either all the estimates of an analyst are anonymized, or only the most recent (so that only his direct clients have access to the complete information). This article remarked that past forecasts are also anonymized, especially bad forecasts, thereby invalidating the use of the database to rank analysts.

Using Economic and financial information for stock selection
I. Roko and M. Gilli (2006)

Instead of building a model using linear regression on a moving window, they use a *classification tree*.

They do not produce a single tree, but a set of trees (a *bag*, a *forest*), each estimated on a bootstrap sample, to lower the variance of the predictor – this should be reminiscent of the set of models produced by bayesian model averaging.

They do not predict the future returns of the stocks but just whether they outperform, underperform or remain neutral – but the method can be generalized to *regression trees* (and forests).

They discretize the predictive variables, thereby discarding information: this is a mistake.

Dynamic market intelligence
M. Krause (2006)

The author suggests using an alpha (valuation, sentiment, quality, risk) to build a portfolio, by investing in the top stocks (his portfolio has the same capbin and sector composition as the market and is equal-weighted in each sector×capbin).

Investment strategies using options on ETFs
A. Seddik Meziani (2006)

On the use of *covered calls* (calls written when you own the underlying, as opposed to *naked calls*): if you think the price will drop for a stock you own, you can write a call: if the price goes up, you just fail to cash on it, if it goes down, you do not lose anything and even pocket the option price.

The author also mentions *short straddles* (write a put and a call, if you think the price will not move) and *long straddles* (buy a call and a put, if you think the price will move but have no view on its direction).

The other January effect
M.J. Cooper et al.
Journal of Financial Economics (2006)

The *January Effect* states that small and low-priced stocks that lose a lot at the end of the year rebound in January.

The (completely unrelated) *Other January Effect* states that January returns help predict the returns of the remaining 11 months. The article confirms this effect, for capitalization-weighted or equal-weighted portfolios, large or small stocks, value or growth stocks.

***Applied Functional Data Analysis
Methods and Case Studies***
J.O. Ramsay and B.W. Silverman
Springer Verlag (2002)

One can reliably smooth a function, defined by a noisy set of points, by expanding it over a basis (e.g., splines – in case of periodic phenomena, you can also consider a Fourier expansion) and by adding a penalty: if you want the k th derivative to be smooth, add a penalty on the $(k + 2)$ th derivative. The smoothing parameter can be estimated by *cross-validation*.

The principal component analysis (PCA) of smoothed data is still noisy and not interpretable: *functional PCA* (fPCA) solves the problem by putting the smoothing inside the PCA algorithm – e.g., for the first component, you maximize a penalized variance. More precisely, the usual conditions

$$\begin{aligned} \forall i & \int \xi_i^2 = 1 \\ \forall i \neq j & \int \xi_i \xi_j = 0 \end{aligned}$$

become

$$\begin{aligned} \forall i & \int \xi_i^2 + \lambda \int \xi_i''^2 = 1 \\ \forall i \neq j & \int \xi_i \xi_j + \lambda \int \xi_i'' \xi_j'' = 0. \end{aligned}$$

Thanks to the use of basis functions, the computations boil down to linear algebra.

To plot and interpret the principal components, just plot the mean function plus and minus some multiple of those components.

The book uses fPCA to study crime careers – but the same ideas could be applied to parallel plots.

Classical methods (e.g., Linear Discriminant Analysis (LDA)) can either be applied on the first functional principal components or “penalized” as PCA was.

A *phase-plane plot*, i.e., a plot of the acceleration against the velocity (since you want to use the second derivative, use a penalty on the fourth), can be interpreted in terms of energy and momentum; one should look for cycles, their radius, their center, changes in their shape, etc. The book examines a consumption index phase plane plot.

To expand plane curves (here, bone shapes) over a basis of functions, one can use *landmarks* or parametrize the curves with *arc-length*.

If the principal components are not directly interpretable, you can perform a *varimax rotation*: this is a rotation in the subspace spanned by the first few principal components that maximizes the variance of the coordinates (“loadings”); it makes the loadings either large or close to zero and therefore eases the interpretation of the components. With the *promax rotation*, one first performs a varimax rotation, takes the power 2 or 4 of the loadings, so as to simplify them further, and looks for an invertible (not necessarily orthogonal) matrix Q that approximates those loadings: $Qb \approx (R_{\text{varimax}}b)^2$. (In R, check the `varimax` and `promax` functions.) This sounds like a poor man’s *independent component analysis* (ICA).

One can use the same ideas to study families of sample distributions, by smoothing the sample distribution functions. Those distributions can then be used in, say, hierarchical models.

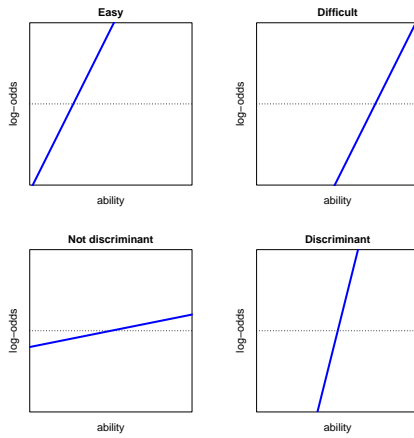
The mean curve may fail to exhibit the features of the individual curves, especially if (as in the example of human growth) they differ in location – they are averaged out. Before computing the mean, you can align the functions: this is called *registration* or (*dynamic*) *time warping*. This accounts for phase changes – but one can also allow for amplitude changes – this can be expressed and implemented as an eigenvalue problem.

Registration transforms the set of curves to study into a set of pairs of curves: the aligned curves and the warping functions $h(t)$ (or the *time deformation functions* $h(t) - t$).

Depending on the data, you may want to register the functions themselves of their first derivative.

Monotone curve smoothing (e.g., human growth or time warping functions) can be performed as follows: instead of looking directly for a smooth increasing function H , look for the *relative acceleration* $w = H''/H'$, i.e., look for a differential equation $H'' = wH'$. As usual, smooth w using a penalty on $\int |w''|$.

Item Response Theory (IRT) studies answers to ability tests. The *ability curve* is a curve in $[0, 1]^N$, where N is the number of questions, whose k th coordinate is the probability of correctly answering the k th question – we assume that the test allows us to rank the candidates, i.e., that they are indeed described by a 1-dimensional subspace, i.e., that the test only measures one thing –, it goes from **0** (bad candidate) to **1** (excellent candidate) and wriggles in between depending on the *difficulty* and *discriminability* of the questions.



The arclength on that curve can be used as a measure θ of *ability*.

IRT models the probability of a correct answer to question i as a function of the ability θ : logistic regression measures difficulty and discriminability, but one can also devise models that account for a non-negligible probability of right answer at low ability (the candidate can answer at random). The picture is complicated by the fact that the ability θ is unknown.

Functional PCA allows more freedom in the modeling of the success probabilities.

A *functional linear model* is a continuous analogue of a regression of a time series versus another, including lags:

$$y_i(t) = \alpha(t) + \sum_{s=t-k}^t \beta_s(t)x_i(s) + \varepsilon_i(t)$$

$$y_i(t) = \alpha(t) + \int_{t-k}^t \beta(s,t)x_i(s) ds + \varepsilon_i(t).$$

The data can be visualized by a *cross-correlation matrix*, the model can be fitted after discretization by *finite elements*, the fit can be assessed by looking at the MSE (mean square error) as a function of time.

$$\text{SSE}(t) = \sum_{i=1}^N (y_i(t) - \hat{y}_i(t))^2$$

$$\text{SSM}(t) = \sum_{i=1}^N (y_i(t) - \bar{y}_i(t))^2$$

$$R^2 = 1 - \frac{\int \text{SSE}(t)dt}{\int \text{SSM}(t)dt}$$

One can characterize a family of functions (here, handwritten letters) by a differential equation

$$D^3 f_i = \alpha + \beta_0 f_i + \beta_1 D f_i + \beta_2 D^2 f_i + \varepsilon_i$$

which can then be used for supervised classification: *Principal Differential Analysis* is a functional analogue of LDA.

This differential equation can actually be a vector one, that allows for *coupling* between the coordinates – especially when there is no canonical coordinate system (here, the hand movements of a juggler).

Those differential equations can be seen as a continuous analogue of regression and the coefficients can be estimated by *integrated least squares*.

Stock market diversity **R. Fernholz, Intech (2005)**

Studying portfolios in a 1-period world can be misleading: by modeling them as stochastic processes (with time-varying, *i.e.*, stochastic, weights), one can decompose the portfolio returns into the contribution of its constituents and the contribution (*excess growth rate*) of the volatility and the correlation of those constituents, accounting for rebalancing and diversification – this is called *stochastic portfolio theory*.

The distribution of the capitalization in the market changes over time; one can measure the *diversity* of this distribution with the *entropy*

$$S = \sum_i -w_i \log w_i$$

where w_i are the market capitalization weights, or with

$$D_p = \left(\sum_i w_i^p \right)^{1/p}$$

(people suggest $p = 0.5$ or $p = 0.76$). Each such diversity measure is canonically associated with a *diversity-weighted portfolio* (in our examples, its weights would be $1/N$ or (proportional to) $w_i^{1/p}$).

One can note that when market diversity rises, active strategies outperform passive ones and conversely.

Diversity measures are linked to the performance of “size” factors.

Diversity-weighted indexing **R. Fernholz, et al., JPM (1998)**

Capitalization-weighted indexes are never rebalanced, except when stocks enter or leave the index, *i.e.*, stocks with a historically high or low price are bought or sold – the opposite of what you should be doing.

Diversity-weighted indexes mitigate this problem: the weights are proportional to some power, $p \in [0, 1]$, of the market capitalization (e.g., $p = 0.76$).

The returns of a diversity-weighted portfolio can be decomposed into a contribution of the variation in the distribution of the capitalization of the stocks and a contribution of the changes in stock rankings (“kinetic differential” – strictly speaking, this component also accounts for dividend payments and stocks leaving and entering the index). The former is noisy and mean-reverting, the latter brings the returns.

Volatility capture as an alpha source **L. Vasquez, SEI (2005)**

Rebalancing a portfolio so that its weights remain constant is better than a buy-and-hold strategy.

“Volatility capture” is also sometimes called *compound growth* or *diversity-weighted indexing*.

Diversification returns and asset contributions
D.G. Booth and E.F. Fama
Financial Analysts Journal (1992)

A portfolio rebalanced so as to keep its weights constant performs better than a non-rebalanced one. More precisely, the returns of a buy-and-hold strategy are:

$$\begin{aligned}\sum_i w_i \left(\prod_t (1 + r_{it}) - 1 \right) &\approx \sum_i w_i (E[r_{i.}] - \frac{1}{2} \text{Var}[r_{i.}]) \\ &\approx \sum_i w_i r_i - \frac{1}{2} \sum_{ij} w_i \sigma_{ii}\end{aligned}$$

while those of a rebalanced one are

$$\begin{aligned}\prod_t \left(1 + \sum_i w_i r_{it} \right) - 1 &\approx E \left[\sum_i w_i r_{i.} \right] \\ &\quad - \frac{1}{2} \text{Var} \left[\sum_i w_i r_{i.} \right] \\ &\approx \sum_i w_i r_i - \frac{1}{2} \sum_{ij} w_i w_j \sigma_{ij}.\end{aligned}$$

The difference between the two is called the *excess growth rate*:

$$\frac{1}{2} \left(\sum_i w_i \sigma_{ii} - \sum_{ij} w_i w_j \sigma_{ij} \right).$$

Stochastic portfolio theory and stock market equilibrium
R. Fernholz and B. Shay
Journal of Finance (1982)

A continuous-time derivation of the *excess growth rate*.

Model uncertainty and forecasting, a practitioner point of view
B. Bellone and E. Michaux (2006)

The article compares several model selection procedures:

- Linear regression bayesian model averaging (BMA) is good, especially for the short term;
- Dynamic factor models (DFA), *i.e.*, regression on the factors from a factor analysis and their lags, is not bad, especially for the long term.

The article also examined

- Median BMA (the model obtained from the variables whose inclusion probability is beyond 1/2);
- The general-to-specific (GETS) algorithm (a step-wise variable selection, with multiple steps and tests).

The authors use a Scilab econometric toolbox developed at the ENSAE: “Grocer”.

Portfolio insurance: the extreme value approach to the CPPI method
P. Bertrand and J.L. Prigent (2001)

The CPPI (constant proportion portfolio insurance) method is a simplified asset allocation strategy that tries to guarantee a minimum terminal wealth. The *floor* is the present value of the insured terminal value; the *cushion* is the difference between the portfolio value and the floor (it should remain positive). CPPI suggests to invest a fixed proportion (the *multiple*, e.g., 10) of the cushion in the risky asset and keep the rest in the risk-free asset – if the cushion is already large, 10 times the cushion could exceed the portfolio value: in this case, everything would be invested in the risky asset.

The article explains how to choose the multiple, using quantiles.

Options-based portfolio insurance (OBPI) is another portfolio insurance (PI) method.

Financially-motivated model performance measures
C. Friedman and S. Sandow
Journal of credit risk (2006)

Classical performance measures (e.g., contingency table-based measures such as the type I and type II error rates; rank-based measures (that fail to properly account for extreme events); calibrated measures (?); conditional information entropy ratio (CIER) (?), etc.) do not easily generalize beyond two states and are not consistent with utility theory. The article suggests to compare the performance of two models with their Kullback–Leibler distance to the data:

$$\text{KL}(\text{model 1, data}) - \text{KL}(\text{model 2, data}).$$

Those Kullback–Leibler distances can be expressed in terms of a utility function, to be chosen – preferably of the form

$$U(\text{wealth}) = \frac{\text{wealth}^{1-\kappa} - 1}{1 - \kappa}.$$

This setup can be robustified and generalized to multi-state or continuous models.

A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity
H. White, Econometrica (1980)

In presence of heteroskedasticity, the coefficients of the linear model $y \sim x$ are consistent, but their variance matrix, usually defined as

$$\frac{1}{n} \sum_i \hat{\sigma}^2 x_i' x_i$$

is not. Instead, one can replace the standard deviation $\hat{\sigma}$ by the residuals $\hat{\varepsilon}_i$:

$$\hat{V} = \frac{1}{n} \sum_i \hat{\varepsilon}_i^2 x_i' x_i.$$

V. DeMiguel et al. (2005–2006)

If you do not have much data, assign the same fraction of your wealth to each asset in your universe; if you have more data, use a minimum-variance portfolio; only if you have huge amounts of data (years of tick data, centuries of daily data) should you consider mean-variance optimization. In all cases, imposing constraints to your optimization problem is a good idea.

Towards a new early warning system of financial crises

M. Bussiere and M. Fratzscher
Journal of international money and finance
(2006)

Early warning systems (EWS) of financial (currency) crises for emerging market economies (EME) should not consider two states (crisis, non-crisis) but three (pre-crisis, crisis, post-crisis), to avoid the *post-crisis bias*.

The january effect

M. Haug and M. Hirschey
Financial Analysts Journal (2006)

The *January effect* (small capitalizations that experienced large losses the previous year have high returns in January) is still there.

How profitable is capital structure arbitrage?

F. Yu
Financial Analysts Journal (2006)

If you need an introduction to capital structure arbitrage (CSA) and the CreditGrades model...

The conclusion of the article is that CSA is profitable, unless you “forget” to build a diversified portfolio and focus on a single company – this is, unsurprisingly, too risky.

Stock price reaction to public and private information

C. Vega
Journal of Financial Economics (2006)

We already use the dispersion among the analyst estimates as an investment signal: this article claims that dispersion in the news (measured by the amplitude of the subsequent excess returns) is also an important signal, more important than the number of news items or the private-information-based trading (PIN).

Is information risk priced for NASDAQ-listed stocks?

K.P. Fuller et al. (2007)

The PIN (probability of informed trading – they recall the formula but do not provide any insight) work on the NYSE but not on the NASDAQ.

The copula-GARCH model of conditional dependancies: an international stock market application

E. Jondeau and M. Rockinger
Journal of international money and finance
(2006)

There are several models describing the evolution of correlation matrices (multivariate GARCH, CCC, etc.) but they do not fully account for fat tails. The copula-GARCH model introduced in the article allows for Student innovations and copulas to capture non-elliptical distributions and time-varying higher moments (coskewness, cokurtosis).

Are our FEERs justified?

G. Barisone et al.
Journal of International Money and Finance
(2006)

To gauge whether current exchange rates are at equilibrium or if some currencies are too expensive or cheap, one often uses purchase power parity (PPP) models, such as The Economist’s Big Mac Index. The fundamental equilibrium exchange rate (FEER) method uses import/export volumes/prices. Contrary to PPP, the resulting exchange rates are cointegrated with the actual exchange rates.

Axioma’s alpha factor method
Axioma (2006)

The alpha factor method is a modification of the definition of the risk of a portfolio with respect to a risk model that tries to account for risk underestimation. For instance, optimizers can tell you that 0% of the risk is due to the size factor, which is almost always wrong. This is due to the “null space of the exposures matrix” being very large. The alpha factor accounts for the risk hidden by the closeness to this null space.

$$\text{risk}^2 = w'(EVE' + ff' + \Delta)w$$

$$f = \sigma \frac{Pw}{\|Pw\|}$$

$$P = I - B(B'B)^{-1}B'$$

(Here, P is the orthogonal projection onto $\ker B'$ and \cdot^{-1} is a generalized inverse.)

Hybrid risk models (*i.e.*, risk models with added *statistical factors*) are another way of accounting for those missing phantom factors, but the alpha factor method is adapted to the portfolio – and since the portfolio changes during the optimization process, the alpha factor changes as well.

Their optimizer allows for several benchmarks and can therefore be used to combine several alpha components: first build a *pure portfolio* for each alpha component, with no constraints; then build a portfolio, with constraints, and control the tracking error to each of the pure portfolios.

***The earnings quality skew
and long-short investing***

J.J. Mezrich and J. Feng, Nomura (2006)

With the alpha used in long-short strategies, e.g., earnings quality, the long and short performance are usually not synchronized – this can be referred to as an *alpha skew*. One can tap into that phenomenon by optimizing a two-asset portfolio made of a long-vs-market pair and a market-vs-short pair (requiring that the weights be positive and that the maximum weight be 1, or whatever your leverage is). This can be seen as “shrinking” either the long or the short portfolio towards the market portfolio. Those dynamic weights exploit the full variance matrix between the excess long returns and the excess short returns – this includes, but is not limited to (as the article claims), their correlation.

The article fails to compare this approach with a full-blown optimization, over the whole universe: is it more than a poor man’s optimized portfolio?

Edgeworth binomial trees

M. Rubinstein

Journal of derivatives (1998)

Option pricing need not be confined to the log-gaussian realm: one can devise “binomial trees” that match prescribed mean, standard deviation, skewness and kurtosis and use them to price options.

The *Edgeworth expansion* of a univariate distribution is a tractable approximation whose first four moments match.

This sounds very similar to the Cornish-Fisher value at risk.

The strategy of professional forecasting

M. Ottaviani et al.

Journal of financial economics (2006)

Analysts can have two goals:

- be the best – they then formulate bold forecasts: if their forecast is not the best, how wrong they are is irrelevant, so they try to have as few competitors as possible;
- be good, most of the time – they then shrink their forecasts towards the consensus, in order to take as few risks as possible.

***Political relationships, global financing
and corporate transparency:
evidence from Indonesia***

C. Leuz et al.

Journal of financial economics (2006)

The authors measure the closeness of a company to the current president by looking at the returns on the days presidential health problems were announced.

Divergence of opinion and equity returns

J.A. Doukas et al.

**Journal of financial and quantitative analysis
(2006)**

Stock returns are positively associated with divergence of opinion – not negatively as previously thought.

***Stock returns, implied volatility innovations
and the asymmetric volatility phenomenon***

P. Dennis et al.

**Journal of financial and quantitative analysis
(2006)**

The *asymmetric volatility phenomenon* (AVP) is yet another stylized fact: negative return shocks are followed by a higher volatility than positive returns.

Actually, those shocks do not affect the idiosyncratic volatility, but only the systematic one.

***Multivariate market association
and its extremes***

D. Baur (2006)

Besides the dispersion, *i.e.*, the cross-sectional (target) standard deviation, one can also consider the upward or downward dispersion.

A simple framework for time diversification

F.J. Fabozzi et al.

Journal of investing (2006)

The *time diversification index* (TDI) is

$$\frac{\text{Short-term Sharpe Ratio}}{\text{Long-term Sharpe ratio}}$$

where the (generalized) Sharpe ratio is

$$\text{Sharpe ratio} = \frac{\text{Risk}}{\text{Expected returns}}$$

for some measure of risk (standard deviation, value at risk, etc.)

***Using neural networks to analyze
intermarket relationships***

L.B. Mendelsohn

Journal of trading (2006)

The author suggests to look at momentum not only backwards (previous returns) but also sideways (returns in other markets), for instance using neural networks.

***Let's play hide-and-seek: the location and size
of undisclosed limit order volume***

S. Bongiovanni et al.

Journal of trading (2006)

Predicting hidden order volume in high-frequency data (we do not only use the trades but also the quotes).

***A fresh look at
investment performance evaluation***

R.J. Surz

Journal of Portfolio Management (2006)

The author advocates the use of random portfolios.

Are optimizers error maximizers?

M. Kritzman

Journal of Portfolio Management (2006)

Actually, no – the profile (returns, risk, exposures, etc.) of the correct and incorrect portfolios are very similar.

Smoothly mixing regressions

J. Geweke and M. Keane (2005)

One can extend the mixture of gaussians model by allowing the probabilities to depend (via a probit link) on covariates.

***The information in option volume
for future stock prices***

J. Pan and A.M. Poteshman (2006)

An options-based order flow imbalance (OFI): the put-call volume ratio.

***Portfolio optimization
with robust estimates of risk***

V. DeMiguel and F.J. Nogales (2006)

Mean-variance portfolios, or even minimum variance portfolios, are very unstable. One can replace the 2-step process “estimate the risk model then compute the minimum variance portfolio” by a 1-step process (just put the formula for the variance matrix in the objective function) and then replace the squares that appear in the objective function by a robust loss function (either slowly increasing, such as the *Huber function*, leading to *M-estimators*; or bounded, such as the *biweight function*, leading to *S-estimators*).

***Beauty contests and
iterated expectations in asset markets***

F. Allen et al.

Review of Financial Studies (2006)

The law of iterated expectations (today’s expectation of tomorrow’s expectation of the price in two days equals today’s expectation of this price) holds for expectations given the public information, for expectations given the information available to a given investor, but does not hold for the average expectation over all investors – and those average expectations play a central role in establishing future prices.

***Estimation of approximate factor models: is it
important to have a large number of variables?***

C. Heaton and V. Solo (2006)

Principal Component Analysis (PCA) decomposes a sample variance matrix as $V = EE'$ (while factor mod-

els try to write it as $V = EE' + \Delta$, where Δ is diagonal); this decomposition converges to the population PCA when $T \rightarrow +\infty$ with N fixed. This article examines what happens when T and N both tend to infinity with the ratio N/T fixed.

Who herds?

D. Bernhardt et al.

Journal of Financial Economics (2006)

Analysts do not shrink their forecasts towards the consensus – on the contrary. The absolute value of the forecasts is irrelevant: focus on the forecasted value relative to the consensus.

A conditional approach to hedge funds

F. Pochon and J. Teiletche

Journal of Alternative Investments (2006)

Fit a *regime-switching* model (mixture of gaussians, estimated with the EM algorithm) to a chosen *core asset* (e.g., the S&P 500 index, if you are interested in equities) to identify two market regimes: quiet and hectic. You can then compute exposure, correlation, etc., conditional on the market regime, *i.e.*, you have two values for each measure, a quiet and a hectic one. You can also perform (simulation-based) statistical tests to check whether the regime makes a difference.

***How sub-optimal – if at all –
is goal-based asset allocation?***

J.L.P. Brunel

Journal of Wealth Management (2006)

A behavioral (or goal-based) portfolio is a combination of efficient portfolios, each corresponding to a different goal, *i.e.*, to a different risk aversion. But a combination of efficient portfolios need not be efficient – how inefficient is it? Not much, actually.

***Optimization and quantitative
investment management***

A. Khodadadi et al.

Journal of Wealth Management (2006)

We need a decent IT infrastructure, with an easy to access and maintain database – a *data warehouse* (DW) – and an optimizer.

***Optimal portfolio allocation
using funds of hedge funds***

J.-P. Gueyie et al.

Journal of Wealth Management (2006)

Hedge fund (HF) indices over-estimate their returns, mainly because of *survivor bias* – probably 3% per year. This problem is mitigated by *funds of funds* (FOF).

The article presents several risk measures, adapted to hedge funds, and runs portfolio optimizations with them:

– *Standard deviation* (one can also consider skewness and kurtosis, which are not risk measures, but can

complement the standard deviation to build other risk measures);

- *Expected loss* and Ω ;
- *Semi-variance*: $E[(X - EX)^2 | X < EX]$;
- *Target semi-variance*: $E[(X - X_0)^2 | X < X_0]$, for some user-chosen X_0 , e.g., $X_0 = 0$;
- *Maximum drawdown*;
- *Value-at-Risk* (VaR), i.e., a quantile, that can be computed empirically, empirically with exponentially-decaying ($\alpha = .99$) weights, assuming a gaussian distribution, using a *Cornish-Fisher expansion* (a kind of Taylor expansion of the quantile function of a distribution, around the gaussian quantile, that involved excess skewness S , excess kurtosis K and the corresponding gaussian quantile z)

$$\text{CFVaR} = \sigma \left(z + \frac{z^2 - 1}{2} S + \frac{z^3 - 3z}{24} K + \frac{2z^3 - 5z}{36} S^2 \right)$$

or using *extreme value theory* (EVT).

To select a portfolio on the efficient frontier for one of those measures you can use the *modified Sharpe ratio*

$$\text{MSR} = \frac{\text{Expected returns}}{\text{Chosen risk measure}}.$$

The conclusion of the article is that hedge funds (or funds of funds) are good, but non-gaussian.

A data-driven optimization heuristic for downside risk minimization **M. Gilli, Journal of Risk (2005)**

This article advocates *threshold accepting* (TA), a variant of simulated annealing (replace the probability by a threshold; select the cooling scheme from the empirical distribution of the differences of the objective function), together with downside risk measures such as value at risk (VaR), expected shortfall (ES) and Ω .

$$\begin{aligned} I_1 &= \int_{-\infty}^0 F(z) dz \\ I_2 &= \int_0^{\infty} (1 - F(z)) dz \\ I_3 &= \int_{\text{VaR}(\alpha)}^{\infty} (1 - F(z)) dz \\ \text{ES}(\alpha) &= \text{VaR}(\alpha) + \frac{1}{\alpha} I_3 \\ \text{EL} &= I_2 - I_1 \\ \Omega &= I_2 / I_1 \end{aligned}$$

Comparing downside risk measures for heavy-tailed distributions **J. Danielsson (2005)**

For random variables *with regularly varying tails*, i.e., their tail can be approximated by a Pareto distribution, i.e.,

$$\forall x > 0 \quad \lim_{t \rightarrow -\infty} \frac{F(tx)}{F(t)} = x^{-\alpha},$$

i.e., $F(-x) \underset{x \rightarrow +\infty}{\sim} Ax^{-\alpha}$, expected shortfall does not depend (much) on the tail coefficient A but only on the tail index α , which is bad news for financial applications, where the tail index α is the same across stocks but where the tail coefficient A varies. Other downside risk measures (VaR, semi-variance, first lower partial moment, zeroth lower partial moment) do not have this problem.

Modeling model uncertainty **A. Onatski and N. Williams (2002)**

To account for model uncertainty, say, a regression preceded by a variable selection procedure, you can:

- Forget about the bias introduced by variable selection and compute prediction intervals; if the model is more complicated than linear regression, use bayesian methods to get the distribution of the estimators;
- Consider the set of models (i.e., selected variables) containing the chosen model, with suitable weights;
- Use the set of all models.

This article applies those ideas to State Space Models (SSM) and uses them to make the best decision in the worst case (minimax).

Mining source code elements for comprehending object-oriented systems and evaluating their maintainability **Y. Kanellopoulos et al.**

Clustering source code metrics of Java classes (number of lines, number of methods, number of public methods, number of children, tree depth, number of coupled classes) can help us understand (and assess the maintainability of) large software projects.

The problem of disguised missing data **R.K. Pearson (SIGKDD Explorations, 2006)**

Beware of missing data; beware of missing data not labeled as such (outliers or even inliers); look at the data.

Graph-theoretic scagnostics **L. Wilkinson et al. (2005)**

Scatterplot matrices can be insightful, but when you have dozens or hundreds of variables, the scatterplot matrix is unreadable: on which cells of this matrix should we focus? To answer this question, one can devise several measures of “interestingness” or “peculiarity” of a scatterplot: presence of outliers, skinniness, curvature, skewness, clumpiness, etc.; then, one can look at the scatterplot of those peculiarity measures (the variables are the interestingness measures and the

observations are the cells of the initial, large scatterplot matrix, *i.e.*, the pairs of initial variables) and select, interactively (this is called *brushing*) the outliers and look at the corresponding scatterplots.

Those measures are sometimes called *scagnostics*.

Classical scagnostics are parametric and/or do not lend themselves to large data sets: this article provides non-parametric, algorithmic, graph-based scagnostics, suited to situations with 100,000 observations and 100 variables.

***After-tax asset allocation*
W. Reichenstein (FAJ, 2006)**

You learnt not to mix present value (PV) and future value, and to convert everything to PVs – you should do the same with taxes, and convert everything into after-tax value.

***Covariance misspecification in asset allocation*
S.P. Peterson and J.T. Grier (FAJ, 2006)**

Estimates covariance matrices can be “wrong” (biased and overly noisy) for two reasons:

- Not all the assets have the same history length: people usually truncate the time series to match the shortest, but this discards information and is suboptimal: for instance, we lose precision on the variance of the longer time series;
- Some return series (e.g., real estate, private equity) are smoothed: their variance will be biased towards zero.

The article recalls that covariance matrix estimators allowing for missing values do exist (Stamgaugh variance) and that unsmoothing methods do exist (Fisher–Geltner method – actually a special case of ARMA innovations) – details are lacking, though.

***Gradient maximization: an integrated return/risk portfolio construction procedure*
J.S. Brusk and V.K. Schock (2006)**

Portfolio construction is usually a two-step process: first predict the returns of all the stocks, then build a portfolio. This can be replaced by a one-step process. A straightforward (linear) implementation proves more robust to non-linearities.

***Holistic asset allocation for private individuals*
G.B. Fowler and V. de Vassal, Journal of wealth management (2006)**

Private investors often allocate their assets to goals (“retirement”, “charity”, “next generation”, etc.) and typically partition their total assets into separate buckets for those goals.

It is preferable not to separate things and to state the client requirement in a single (more complicated) optimization.

***Optimal rebalancing strategy for institutional portfolios*
W. Sun et al. (2006)**

Dynamic programming-based rebalancing performs better than scheduled rebalancing or fixed tolerance band rebalancing – it is adaptive.

The article compares the three strategies with several utility functions (quadratic, log wealth, power) and examines their robustness.

Dynamic programming (you should know what it is before reading this: the article is too technical) is not amenable to current computers beyond a dozen assets.

***Parameter estimation of ARMA models with GARCH/APARCH errors: an R and SPlus software implementation*
D. Würtz et al. (2006)**

If you want to understand how GARCH (or APARCH) is implemented in R, or if you want to modify the code to accommodate other models.

***Dynamic forecasting behavior by analysts: theory and evidence*
J. Clarke and A. Subramanian
Journal of financial economics (2006)**

Significant underperformers or outperformers are more likely to issue bolder forecasts.

***Stock returns, implied volatility innovations and the asymmetric volatility phenomenon*
P. Dennis, S. Mayhew and C. Stivers
Journal of financial and quantitative analysis (2006)**

The relation between stock returns and volatility innovations is mainly due to the market; can be refined by including the market previous returns; the volatility innovations can be proxied by the implied volatility innovations.

***A portfolio of stocks and volatility*
R.T. Daigler and L. Rossi
Journal of Investing (2006)**

Volatility (e.g., VIX futures, variance options, swaps) is negatively correlated with the S&P 500 and should therefore be included in equity portfolios.

But beware: as investors are getting more familiar with options, the usefulness of options-based strategies change – in your simulations, do not assume that the situation is stationary.

***Improving risk-adjusted returns of fixed-portfolios with VIX-derivatives*
G. Dong (2006)**

As above.

***A tactical implication of predictability:
fighting the FED model***

R. Salomons, *Journal of Investing* (2006)

The *FED model*, that selects between equity and bonds by comparing earnings yield and bond yield, is not valid: it forgets inflation (*money illusion*) and that the relation between bonds and equity is not constant over long stretches of time.

The author suggests to regress earnings yield against bond yield and a few more variables such as equity volatility and bond volatility.

useR! 2006 conference

Last week, I attended the 2006 useR! conference: here is a (long) summary of some of the talks that took place in Vienna – since there were up to six simultaneous talks, I could not attend all of them...

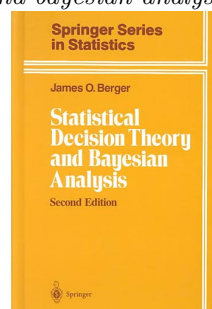
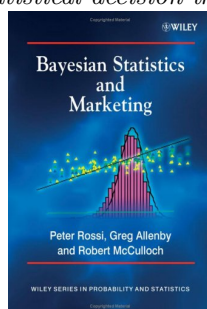
0. General remarks

There were 400 participants, 160 presentations.

Among the people present, approximately 50% were using Windows (perhaps less: it is very difficult to distinguish between Windows and Linux), 30% MacOS, 20% Linux (mostly Gnome, to my great surprise).

1. Tutorial: Bayesian statistics and marketing (Peter Rossi)

The goal of statistical inference is to make probability statements from various information sources: the data, but also *prior* sources, for instance, “this parameter should be positive” or “this parameter should have reasonable values”. The difference between marketing and econometrics is that in the latter, those statistical statements lead to actions – see J. Berger’s book, *Statistical decision theory and bayesian analysis*.



Bayesian methods produce the whole posterior distribution of the parameters, from which you can extract any information – not simply “the most likely value” of this parameter, as with maximum likelihood (ML) estimators. This is akin to the *sampling distribution*, *i.e.*, the distribution of the estimated (say, ML) parameters if we had run the experiment millions of times.

Bayes’s theorem simply says that the *posterior* probability (or probability density function) is (proportional to) the product of the prior and the likelihood.

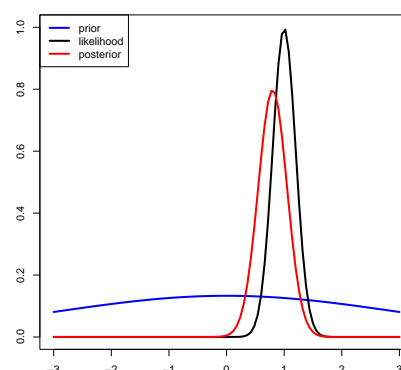
Bayesian statistics can be applied to any kind of model, *e.g.*, binomial, regression, multiple regressions, probit, logit, hierarchical, etc.

The *beta distribution* is a good candidate for parameters in $[0, 1]$: it can be symmetric or skewed, with a large or narrow peak, or even U-shaped.

The *inverted χ^2* distribution is often used as a prior for variances, because it is amenable to computations: it is said to be *conjugate* – but in the computer era, there is no reason to limit ourselves to these.

In the prior distribution, one often has to choose parameters: these are called *hyper-parameters*.

Bayesian estimators are often said to be *shrinkage estimators*: they are “between” the prior and the maximum likelihood estimators (MLE).



The larger the sample size, the smaller the influence of the prior.

Sampling from multivariate distributions can be tricky: it is often easier to sample from univariate distributions, *i.e.*, to sample one dimension at a time. The idea of *Gibbs sampling* is to replace sampling from (x_1, x_2) by sampling from $x_1|x_2$ and then $x_2|x_1$. This bears a resemblance to the *EM algorithm* – it might even be exactly the same: for instance, both can be used to fill in missing data.

If we draw the intermediary steps of a Gibbs sampler sampling from a bivariate gaussian distribution, only one coordinate changes at a time and the path is a staircase one.

When using Gibbs sampling, be sure to check the following:

- the autocorrelation and the cross-correlation
- the time series of the sampled parameters: are they stationary?
- plot the bayesian estimators versus the ML ones; plot the bayesian estimators versus the bayesian estimators with a different prior
- run several chains (the sampled time time series are often called *Monte Carlo Markov Chains* (MCMC)) and check if they *mix* (*i.e.*, if they look interchangeable).

Having the full distribution of the parameters allows you to extract a lot of information: *e.g.*, you can investigate the distribution of $X_1 \cdot X_2$ or X_1/X_2 (with stan-

dard gaussian distributions, the latter does not even have moments).

In the bivariate gaussian example, since X_1 and X_2 are correlated, successive values of (X_1, X_2) will be correlated: the samples do not contain as much information as an independant sample of the same size.

With the probit model,

$$\begin{aligned} Z &= X\beta + \varepsilon \\ Y &= Z > 0 ? 1 : 0 \\ \varepsilon &\sim N(0, 1) \end{aligned}$$

(this can be seen as a *censored* (or truncated) model: Z is replaced by the intervals $(-\infty, 0]$ or $(0, +\infty)$ – there is less information in censored data than in complete data), bayesian methods provide more information: we also have the distribution of the *latent* (or hidden) variable Z – more generally, bayesian methods provide an estimation of the distribution of the parameters, latent variables and missing values.

Mixtures of gaussians can be tackled in the same way: the latent variable is the number of the cluster. But there, the situation is much worse: a permutation of the numbering of the components does not change anything – as a result, if there are n components, the likelihood has $n!$ modes... Actually, it is not a problem: the Markov chain will *switch the labels* from time to time, but this will have no consequence.

The *multinomial probit* model (the variable to predict is not binary but can take n values – say, margarine brands) is almost intractable with classical methods but is amenable to bayesian methods. Some care is needed, though: see the book.

The same goes for the *multivariate probit* model (the variable to predict is a subset of those n values – say, beer brands).

The *Metropolis algorithm* (an alternative to the Gibbs sample, that allows you to easily sample from a multivariate distribution, and which is preferable when the variables are too dependant) was not tackled – see the book.

Finally, the interesting part: *panel data* and *hierarchical model*. Since this is getting more intricate, I prefer to refer you to the book.

The presenter developped the **bayesm** package, that provides, among others, the following functions:

- **runireg** samples from the posterior distribution of the parameters of a regression $y \sim x$ with an inverted χ^2 prior on the variance σ^2 and a gaussian prior of $\beta|\sigma$
- **rmultireg** samples from the posterior distribution of the parameters of a family of regressions $Y \sim x$ with an *inverted Wishart* prior on the variance Σ and a gaussian prior on $\beta|\Sigma$
- **rbiNormGiggs** samples from a bivariate gaussian, using a Gibbs sampler
- **numEff** computes the effective sample size of a time

series, *i.e.*, the size of the series for independant variables that would contain the same information. It tells you how much thinning you should use.

- **runiregGibbs**: samples from the posterior distribution of the parameters of a regression $y \sim x$ with an inverted χ^2 prior on the variance σ^2 and a gaussian prior of β (and not $\beta|\sigma$)
- **rbprobitGibbs**: samples from the posterior distribution of the parameters of a binary probit regression $y \sim x$ with a gaussian prior on β (and $\sigma = 1$)
- **rmixGibbs**: samples from the posterior distribution of the parameters of a mixture of gaussians with a *Dirichlet prior* on the probabilities of the components, a gaussian prior on their means, an invert Wishart prior on their variance.
- **rmnpGibbs**: samples from the posterior distribution of the parameters of a multinomial probit regression (Beware of the results: they are not intended to be stationary – consider $\beta/\sqrt{\sigma_{1,1}}$ and $\Sigma/\sqrt{\sigma_{1,1}}$ instead)
- **createX**: ancillary function to create the design matrix given to **rmnpGiggs**.
- **rmnlIndepMetrop**: multinomial logit
- **rhierBinLogit**, etc.: there are also a lot of hierarchical models: this is a slippery slope, do not use them unless you *really* know what you are doing.

Conclusion

Mixed models are only partial bayesian methods: you have to provide a prior, but you do not look at the whole posterior distribution, which might be misleading if it not gaussian. Given the power of current computers, there is no need for such a restriction: we can afford a full bayesian method.

There are more general bayesian packages (*Bugs*, *JAGS*), with which you can simulate any kind of model, but that generality comes at a price: on special cases, the computations are not as fast as they could be – by several orders of magnitude...

2. Tutorial: Rmetrics (Diethelm Wurtz)

Rmetrics is a set of R packages for quantitative finance and econophysics, initially developped for educational purposes. This tutorial reviewed those packages one at a time.

The **fBasics** package helps study the stylized facts of financial time series.

The **fCalendar** package is devoted to time manipulations. The notion of *time zone* (TZ) is replaced by that of *financial center*, that encompasses daylight saving time (DST) rules and holidays (to perform operations such as “next business day”). This is still imperfect: different markets in the same financial center have different holidays (e.g., Chicago/Equities and Chicago/Bonds).

The **timeSeries** package defines the `timeSeries` class: those objects contain one or several time series, having the same set of timestamps.

The **fSeries** package provides the **garchFit** function, for fit GARCH models and their variants. GARCH

models are a bit of a problem with statistical software: for a long time, there has been no *benchmark* against which to assess an implementation, yielding very disparate results across systems (*Ox* is not that good, but the others, including SPlus and SAS, are much worse).

The **fExtreme** package is devoted to extreme values. It contains a set of functions to (visually) study distribution tails (`emdPlot`, `lilPlot`, `mePlot`, `msratioPlot`, `qqPlot`, `recordsPlots`, `sllnPlots`, etc.)

To compute the *Value at Risk* (VaR) or the *Expected Shortfall* (ES) one can try to fit a distribution to the tail of the data, chosen from the family of limit distribution of tails of distributions: the *Generalized Pareto Distribution* (GPD).

```
pgdriskmeasures(gpdFit(
  x,
  threshold = .95,
  method = c("pwm", "mle", "obre")
))
```

(Here, `obre` stands for “optimally biased robust estimator”).

Extreme Value Theory (EVT) also studies the distribution of the maximum of iid random variables: there is a limit theorem, similar to the central limit theorem (with `max` instead of `mean`) that identifies the limit distribution as one of the GEV (*Generalized Extreme Values*) distribution (of which the Gumbel, Frechet, Weibul are special cases).

The **fCopulae** package is devoted to *copulas*. The implementation is more reliable than that of SPlus (SPlus seems to use numeric differentiation, which is unstable in extreme cases; Rmetrics uses formal derivatives, computed in Maple).

Copulas address the following fallacies.

- Fallacy 1: Marginal distribution and their correlation matrix uniquely determine the joint distribution.
- Fallacy 2: $\text{Var}(X_1 + X_2)$ is maximal when $\text{Cor}(X_1, X_2)$ is maximal
- Fallacy 3: $\text{Cor}(X_1, X_2)$ small implies that X_1 and X_2 are almost independant.

The **fOptions** package is the best-known part of Rmetrics: all the (equity-based) options, priced by exact formulas (when available), binomial trees, Monte-Carlo simulations (with *antithetic variables*, *low discrepancy sequences*), PDEs.

The **fBonds** package is devoted to bonds (but I am not familiar with bonds).

The **fBrowser** is an Rcmdr-based GUI that provides the above functionalities. You can extend it and add your own menus.

Conclusion: The coverage is impressive, and Rmetrics should be considered if we plan to use options or if/when we start to investigate *Econophysics*.

You may also want to have a look at the Rmetrics

website (Diethelm also has a company, Finance Online Gmbh).

3. Recurring topics

The following topics were tackled in several talks.

GUI

Windows users are typically intimidated by the almost empty starting screen of R and wonder “where is the GUI?”.

Novice users, who do not want to tamper with the command line, are probably better off with an Rcmdr-like interface: indeed several projects build on John Fox’s Rcmdr (which provides basic statistics) to provide domain-specific functionalities with a menu-driven interface: `fBrowser` in Rmetrics for finance, `GEAR` for econometrics, etc.

Programmers also complain about the difficulties of debugging R code and the lack of a VisualStudio-like IDE (Integrated Development Environment). Note that SPlus is addressing this concern by providing an Eclipse-based workbench – there is an Eclipse plug-in for R, but many features are still missing.

Let us also mention ESS, Texmacs and JGR.

Interactive graphics

Some people claim that the lack of interactive graphics is one of the major drawbacks of R: they would like to be able to have several plotting windows, presenting different views of the same data set, to be able to select points in one plot and see the corresponding points highlighted in the others (this is called *brushing*).

iPlots (built with `rJava`) provides those facilities, together with a (portable) GUI, but is still under development – it seemed perfectly useable, though.

GGobi can already do all that, but it is a separate application (that can talk to R) and it does not provide user-defined plots.

The *rgl* package leverages *OpenGL* (one of the technologies used by the graphics card found in most computers and needed to play most video games – a large, untapped source of computational power) to produce 3-dimensional plots, that can be interactively rotated; but their elements cannot be selected, brushed, etc.

Of course, one can still use Tk widgets (and the `tkrplot` package) to produce plots that are automatically updated when the user moves a slider.

e-Learning and collaborative documentation: Wikis galore

Though the R documentation is often better than that of other software, it still has a few problems: the manual pages are terse reference manuals, often unsuitable to begining users; the contributed manuals, that cater to users with very specific backgrounds, are not updated as timely as R is.

To tackle this problem, some suggested to write collaborative documentation, in the spirit of Wikipedia:

after several months of discussions on the R-SIG-Wiki mailing list (about which wiki engine to use, how to have it understand R, which structure the site should have, etc.), Philippe Grosjean opened the R Wiki as the conference started, with already some contents (R tips, the R Tk tutorial, and the first chapter of my *Statistics with R*).

Wikis are also used in several e-Learning projects: the teacher sets up the structure of the site and the students fill it in, with the notes they have taken, the statistical analyses they have carried out, the problems they have had, how they have solved them, etc.

Some companies (Microsoft, IBM, GM, Dresdner, etc.) are using a *corporate wiki* for their intranet and part of their web site – in particular large companies, when they want to show that their employees are human, that their projects are steadily progressing. (If you want another buzzword for that, you can use the more general term *Web 2.0*.)

Frank Harrell suggests to use Wikis for *knowledge management* – his web site is a wiki, and has been so for years. He also encourages us to use Sweave for *document management*.

For more about Wikis, CMS (*Content Management Systems*), and web sites in general, check, for instance GNU/Linux Pratique HS 5 (in French).



Reproducible results

Several people advocated the need for reproducible results, mainly with *Sweave*, and indeed, most of the presentations were made with Sweave and *Beamer*.

Some people (Dirk Eddebuettel) even suggested to cryptographically *fingerprint* the datasets used (do not do that in the US, though: someone managed to patent it – you can fingerprint files, but not files containing data).

Enterprise Business Processes (EBP)

I did not attend all those presentations.

Some explained that R could be used as a component in a larger process, scheduled in an automated way: they usually resort to Rserve or rJava to access R as a web service or as a Java class.

Some of those systems exhibited pretty, impressive but utterly useless (Java) graphical front-ends.

Some explained how to exchange structured data between R and other systems (for simple data, such as a `data.frame`, simply use a CSV file or a database), us-

ing an XML schema (this is sometimes called a DTD) to store `data.frames`, lists, lists of lists, etc. They provide an R package (`StatDataML`) and a Java class (`JStatDataML`) to this end.

(There used to be an XML schema to store and exchange statistical models and data between statistical applications, called PMML, but it was not mentioned and I do not know if the project is still alive.)

Some explained how to extend XSLT to have it call R and perform statistical computations on the data being transformed.

Some explained how to embed R into a web server.

Of course, in this area, the most important thing is the number of acronyms and buzzwords you can fit in a single sentence: as an exercise, try to form a sentence using the words XML, XSLT, JAXB, PyXML, R/Apache, POI, JDBC, Jython, Struts, Hibernate, YAWL, Ruby on Rail.

Large scale computations

When your computational needs grow, you will want to run computations in parallel, on several computers, or several processors on the same machine: these could be completely different processes, similar processes on different data, or a single computation that can be split up into several pieces.

A few packages can facilitate this parallelization: `rpvm` (uses PVM), `rmpi` (uses MPI), `snow` (to transparently parallelize parallelizable code) or `nws`.

The problem of large datasets, that do not fit into memory, was not tackled – the advice did not change, use a database to store the data and/or buy more memory (and use an operating system that can use it).

Unification

There are often several packages on the same subject, each providing similar but different, complementary and incompatible capabilities. In several areas, such as robust statistics (with the `robustbase` package) or econometrics (with the `GEAR` package, that will provide basic econometric functions and a GUI), people are starting to unify all this.

Also note the forthcoming book, *Applied Econometrics with R*, by C. Kleiber and A. Zeileis.

Real-time data, stream processing

One of the challenges faced by R is the increasing amount of data to process and the timeliness of that processing: more and more, we will want real-time results or plots, that pop up as soon as the data arrive, that are updated as soon as the data is.

There is some progress in that direction (such as algorithms to compute a moving median, a moving quantile; or frameworks for enterprise processes that encompass R), but the path to a real stream-processing engine will be long.

(I only attended one of those talks, so I do not know

if the following was mentioned: it is possible to write triggers in R for PostgreSQL and thus launch computations when the data arrive.)

Including R in other software

As non-statisticians progressively want to harness the power of R, they will want to access it from the software they are familiar with, such as spreadsheets or databases.

It is already possible to access R from Gnumeric (a spreadsheet) and from PostgreSQL (a database management system (DBMS)).

(There were also presentations and tutorials about R and Windows, but I did not attend them: they usually assume that you are already a proficient Windows programmer.)

Machine learning

There was a whole session on machine learning, with emphasis on Support Vector Machines (SVM).

Bayesian networks and neural networks were not forgotten, though: an R neural networks toolbox is being developed, similar to the Matlab one.

Bayesian statistics

Bayesian methods rely on two ideas.

First, before doing an experiment or before looking at the data, we have some information: it can be, for instance, a “reasonable” range for the quantities to estimate. This information is called the *prior*.

Second, instead of computing the single “best” value for the parameters of interest, we want the full *sample distribution* of those parameters, *i.e.*, the distribution of the “best” parameters that we would observe if we could repeat the experiment tens of thousands of times: this is the *posterior* distribution.

Those methods used to require lengthy simulations, but the are becoming more and more amenable to commodity PCs.

People have long been using Bugs (WinBugs or the supposedly portable OpenBugs, or its open source replacement JAGS) to sample from the posterior distribution. Those software can accomodate any kind of model, but because of that generality, the computations can take a lot of time.

For very specific models, the computations can be greatly sped up: this is what the *bayesm* and *MCMCpack* packages do – *MCMCpack* also provides you with the building blocks needed to sample from other models.

Bayesian methods can also be used to compare models, as a replacement of *p*-values: check the *BayesFactor* and *PostProbMod* functions in the *MCMCpack* package.

When using bayesian methods, one should not forget to perform a few *diagnostic* tests or plots: this is what the *coda* package does.

Robust statistics

A robust statistical method is one that is not sensitive to outlying data: even if part of the data is outrageously wrong, it has little impact on the results.

This is often measured with the *breaking point* of an estimator (say, a regression): this is the proportion of observations you can tamper with without being able to make the estimator arbitrarily large.

The *influence* of an observation is the change its inclusion or deletion induces in the result.

One can sometimes spot outliers with the *Pearson residuals*:

$$\frac{\text{sample density}}{\text{density according to the model}} - 1$$

(a presenter showed this with circular data, where outliers are not really “far”...).

The *robustbase* package is an attempt to unify the elementary robust methods currently scattered across various packages: it will provide robust regression (*lmrob*, *glmrob*), replacements for the Mean Average Distance (MAD) (*Qn*, *Sn*), MCD (Minimum Covariance Determinant) covariance matrix, etc.

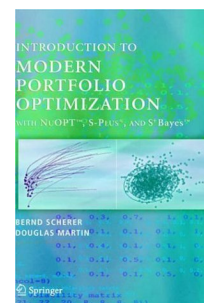
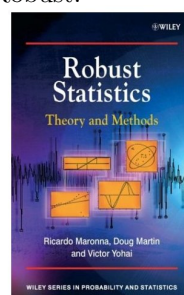
One can “robustify” the (linear, gaussian) *Kalman filter* by replacing the matrix estimations it uses (mainly expected values and covariance) by robust equivalents (median and MCD covariance).

One can robustify Principal Component Analysis (PCA): this is then called *projection pursuit*. PCA finds the direction in which the “dispersion”, as measured by the variance, is largest; robust PCA replaces the variance with a robust equivalent.

One can generalize this to other measures of dispersion, or measures of non-gaussianity (this is called *Independent Component Analysis* (ICA)).

One problem with robust covariance matrices is that their robustness decreases with size (since projection pursuit uses 1-dimensional subspaces, it might not be much of an issue, but for other applications, it is).

There is a mailing list devoted to robust methods in R: R-SIG-Robust.



Regularization paths

A *shrinkage* estimator is an estimator somewhere on the path between a prior estimation (very stable, reliable, but hardly informative) and an estimator (e.g., a MLE estimator: it contains all the information, but can be extremely noisy).

The “best” position on the path can be chosen by *10-fold cross-validation* (CV).

There are variants of this idea:

- *Principal component regression* is a regression in the first k principal components (the path is discrete, indexed by the number of components retained).
- *Forward variable selection* (here again, the path is discrete and corresponds to an order on the set of variables)
- *Ridge regression* is a regression with an L^2 penalty on the amplitude of the coefficients (if some of the predictors are correlated, the corresponding coefficients can be extremely large, with opposite signs: the penalty tries to avoid this)
- *Lasso regression*: idem with an L^1 penalty
- *Forward stagewise regression*: instead of completely adding the variables, as in forward variable selection, just add a small part of them, say $0.1 \cdot X_i$ (the same variable may be added several times, to increase its coefficient)

Least angle regression (LARS) is very similar to forward stagewise regression:

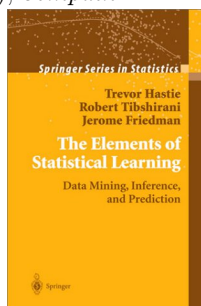
- find the variable X_i the most correlated with the variable to predict Y
- add it in the model, with a small coefficient, and increase the coefficient until another variable, X_j , becomes more correlated with the residuals: then, progressively change the coefficients of the two variables, X_i and X_j , until a third variable...

Lasso and forward stagewise regressions are actually special cases of LARS.

The regularization path of LARS is more stable, less chaotic than that of the lasso.

The number of degrees of freedom if a LARS regression is the number of variables that have been included – with, say, variable selection, it is much more!

There are further generalizations of LARS: *elasticnet* (a mixture of L^1 and L^2 penalties, that tend to select the variables in groups); *glm*path (e.g., for logistic regression); *pathseeker* (take the top k variables instead of the best); *Cosso* (we know, a priori, that the variables are grouped); *svmpath*.



R on Windows and MacOSX

Uwe Ligges tried to convince us that using R on Windows, installing packages from source or even writing you own R packages on Windows was not difficult. He almost made his point: he only needed one slide to list

the prerequisite software (not mentioning how to install them and forgetting about the incompatibilities with other already installed software) and two more slides to explain how to install a package (targeted at advanced Windows users: he tells to change environment variables without reminding us how) – a stark contrast with similar explanations for a Unix platform where, if you do not understand, you simply copy and paste the instructions.

He also noted that using Windows instead of Linux “only” reduced the speed by 10% – which is even more impressive if you consider that 64-bit R on Linux no longer runs slower than 32-bit R on Linux.

However, his talk was followed by a similar talk, by Simon Urbanek, that tried to do the same thing on MacOSX: the differences are amazing (the only instruction is “do not forget to install R”; R is well integrated with other MacOSX applications).

After those two talks, *it really seems insane to use R on Windows* (or anything else than R, for that matter – most of the problems are not specific to R).

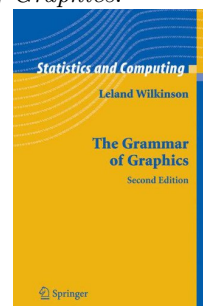
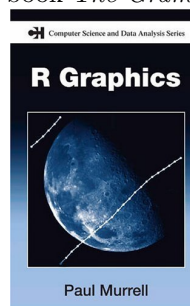
By the way, most of the developers of R (“R-core”, not the people writing the add-on packages, but the people writing the core of the R system itself) are on MacOSX...

4. Other topics

ggplot

You might already know that there are four ways to use Object Oriented Programming (OOP) with R: S3, S4, R.oo and proto.

Similarly, there is now a third way of producing graphics: after the old graphics (with the `plot` function), the lattice graphics (with the `xyplot` and `grid.*` functions), there is now ggplot, that implements the ideas of the book *The Grammar of Graphics*.



forecast

This Windows-only package fits ARMA models to time series, but can infer the order by itself.

Time series modelling

Two talks presented methods to automate the fitting of time series (for instance, with an ARIMA model, you no longer have to select the order(s) of the model).

Multivariate GARCH models

Some multivariate generalizations of the GARCH model were presented, such as *Constant Conditional*

Correlation (CCC: same equation, with diagonal matrices); *Dynamic Conditional Correlation* (DCC: idem, but those matrices are allowed to change over time); *Smooth Transition Conditional Correlation*; *Extended Conditional Correlation* (ECC: the matrices are no longer diagonal, but in order to ensure that the variance matrix is positive definite, you have to add an infinite number of conditions – people usually replace these conditions by a single one, but this is too restrictive); *BEKK*.

All the code presented was developed with a 2-dimensional (or low-dimensional) case in mind.

Particle Filter

A *particle filter* is very similar to a Kalman filter, but it neither assumes that the underlying process is linear nor that the noise is gaussian.

The basic idea is that of an MCMC simulation. Instead of performing 10,000 (independent) simulations, one can try to mix them: at each step, the particles are simply resampled (it sounds trivial, but it is the only difference with an MCMC simulation).

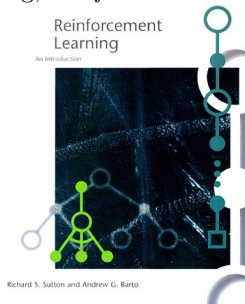
They were applying that to currency data.

Markov Decision Process

It looked interesting, it is related to the use of *dynamic programming* to build portfolios and rebalance them over time, in a multi-period world, but they only had five minutes and I did not understand anything.

They use those ideas to trade currencies.

For more information, google for *Markov Decision Processes* (MDP), temporal distance learning, TD-learning, Q-learning, *reinforcement learning*.



Calibrating the evidence in experiments

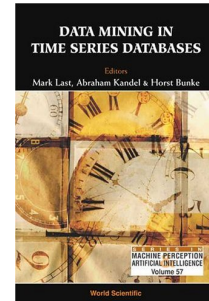
You are probably familiar with my rant against *T*-values (stating *T*-values assumes that your readers are very familiar with the *T* distribution and that they know how the number of degrees of freedom affects those values and does not extend to other tests, you should rather use *p*-values, that simply assume that your readers understand the uniform distribution on $[0, 1]$; the significant level, usually 5%, has a completely different meaning depending on the size of the sample and neither you nor your readers know how to interpret it; you can state a difference of BIC instead of a *p*-value): that presentation (which I did not attend), gives more details about the meaning of the significance level depending on the sample size.

Non-metric clustering

The **riffle** package provides yet another clustering algorithm.

Challenges in cluster analysis

This talk highlighted several areas where cluster analysis is not yet a mature subject: *transaction data* (i.e., clustering subsets, e.g., clustering shopping baskets) and *time series*.



Indian commodity markets

To find a consensus price for a commodity, across a multitude of local markets, in order to set up a futures market, they use an *adaptively trimmed mean*.

Random portfolios

To check if a portfolio manager is a good portfolio manager, one can compare his performance with that of a “random” portfolio: simply generate permuted portfolios from the actual one.

The problem is that these permuted portfolios breach all the constraints the portfolio manager has to abide by. To recover them, one can feed these permuted portfolios to an optimizer, with no alpha and no variance matrix – just the constraints.

According to Patrick Burns himself, the resulting portfolios are not as uniformly distributed as they should be...

If you do not know his company, he sells an optimizer, based on genetic algorithms, that can be called from R (and soon Matlab), and he also does some consulting.

Sparklines

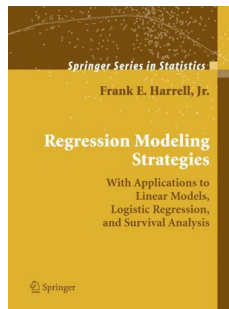
Sparklines are word-like plots, that can be used inside a sentence or in a table.

The dataset is that of the American Statistical Association (ASA) visualization contest.

Barcodeplot and Generalized pairplot

Histograms fail to spot ties in the data: a barcodeplot is similar to a rugplot, but the other dimension is used to indicate ties. It can be seen as a alternative to the boxplot.

This is not unlike the result of the **describe** function, in the **Hmisc** package.



The `pairs()` function in R only accomodates quantitative variables: one can modify it to account for qualitative variables as well, with boxplots or barcode plots when one variable is qualitative and mosaic plots when both are.

Linked micromaps of large financial datasets

When displaying information (here, about mortgages in the UK) on a map, one usually divides it into regions that are coloured according to the average (or total) value of the variables. This is called a *choropleth map*.

But this is misleading: the information conveyed by the plot can change depending on the colour scheme chosen (the boundaries between the colours can be chosen at round values (say, 1, 2, 5, 10, 20, 50, 100, etc.) or at the quantiles of the variable displayed), on the regions chosen (there is a wealth of different, incompatible, administrative (or not) decompositions into regions); larger regions (which tend to have a low density, in our example), are over-emphasized; there is no indication of estimation uncertainty (confidence intervals).

A linked micromap plot is actually a table, with the maps in the first column, and the variables in the others; each row corresponds to four regions, highlighted in the map, with a dotchart in the remaining columns.

See also: <http://www.amstat-online.org/sections/graphics/newsletter/Volumes/v132.pdf>

Impulse response theory (IRT)

(There were several talks about psychometrics.)

Letter-value boxplots

With large datasets, boxplots are not as informative as they should be: in particular, too many points are unduely labeled as “outliers” – such points are supposed to be examined one by one...

Letter-value plots generalize boxplots, by displaying the 1/2, 1/4, 1/8, 1/16, etc. fractiles instead of just the median and the quartile, the zone between two such fractiles is represented by a box, of decreasing width (and or changing colour).

Visualization of multivariate functions, sets and data: denpro

The `denpro` package helps you visualize high-dimensional datasets, as a 2-dimensional plot, stressing its multimodality, its dispersion or its tails.

This looks great, but the presentation was impossible

to understand. The original articles might be easier to read.

Can R draw graphs?

Grid graphics can now draw *X-splines*, connect non-rectangular elements with arrows, clip rectangular regions and (with the `grImport` package) import vector graphics (SVG, PDF, PS – you should transform tem in PostScript first).

Parametric link functions for binary response models

The logit and probit link functions are not always sufficient: tests exist to check if they reflect the data, and if they are rejected, we need to look beyond them.

This talk presented the *cauchit* link function (which is more tolerant to surprising observations), the *Gosset* family (based on tyhe Student T distribution), and the *Pregibon* family (or Tukey lambda family).

Those links are implemented in the `gld` package.

The SPlus package system

If R started as a system “not unlike S” (S in the ancestor of SPlus), the situation is now reversed: to survive, SPlus cannot afford to ignore R.

The next version of SPlus will have a package system very similar to (and compatible with) that of R.

They also try to keep the lead they have in the user interface, by providing an “SPLus workbench” based on Eclipse.

In case you do not know, Eclipse, developped by IBM, is a Java IDE (Integrated Development Environment), *i.e.*, a text editor for Java programmers, that can be extended to accomodate other languages, such as C or C++. There is already an R Eclipse plugin, but it still lacks many features.

Statistical principles to live by (Frank Harrell)

Do not overfit the data: use *shrinkage*, use *penalized likelihood* estimators.

Respect continuous variables, do not bin them before computation.

Use non-parametric methods (not everything is linear).

Account for *multiple tests*: when performing several tests, *i.e.*, when you have several *p*-values, correct those *p*-values.

Be honnest with *multi-step procedures*: you might feel safe when you perform a gaussianity test and decide on the path to follow next (e.g., parametric versus non-parametric tests) depending on the results of this test, but this is actually a multiple test, whose final *p*-value should be corrected – and do not be fooled by the power of non-parametric tests.

Use effective graphics, routinely.

Graphical EDA (Exploratory Data Analysis) using half-space depth

The notion of *depth* of a point in a cloud of points (minimum number of points of the cloud in a half-space passing through that point) can be used to define *bag-plots*, a 2-dimensional generalization of boxplots.

This was the funniest and fastest-paced talk.

Operational Risk management

This talk explained how to use FFT (*Fast Fourier Transform*) to estimate a loss distribution. The data gathering part of their process required a lot of social engineering.

Robustness assessment for composite indicators with R

I did not attend this talk: the author explains how to build a robust (stock) index for a country or a region – this can be seen as a *robust portfolio*.

Capturing non-observed heterogeneity

I did not attend this talk: they cluster discrete time series, using transition matrices.

Random Recursive partitioning

I did not attend this talk: to assess a clustering algorithm, you can apply it on the initial data and on resampled data, and check if the results are similar.

Missing data and Partial Least Squares (PLS)

(not attended)

Supervised self-organized maps

(not attended)

Term structure and credit-spread estimation with R

(not attended)

5. Conclusion

In the forthcoming years, R will face the following challenges:

- real time data processing
- embeddability in other software (spreadsheets, databases)
- large scale computations (distributed or not)

To that list, I would like to add:

- relational data (data that do not fit in a single rectangular table)
- large datasets

Tackling those changes may require drastic changes to R, that will trigger incompatibilities with existing code (the situation could be similar to the switch from Perl 4 to Perl 5, for those of you who lived it).

Game programming gems 6 M. Dickheiser editor (2006)

As the previous ones, this volume contained a few articles that might be relevant outside game development, such as: an explanation of the coding and use of *floating point numbers*; an application of *fuzzy inference*

systems (FIS) to manage scene complexity; the use of *support vector machines* (SVM) on a moving window to implement short-term memory in NPC (non-playing characters); a review of scripting languages (Lua and Python still dominate, but specialized languages are emerging); unit testing in C++ with *CppUnit*; many articles about BSP (binary space partitions) and ABT (adaptive binary trees); parallelism on multicore machines with *OpenMP*; fingerprinting; UDP hole punching (STUN); Python coroutines as *microthreads*; MMOG (but there is not enough liquidity and volatility for a currency manager to step in).

There was also a funny article about “synthesis of realistic idle motion for interactive characters” that managed to combine PCA (principal component analysis), Lie algebras, Markov chains and MST (minimum spanning tree) clustering.

I failed to understand the GPU (shader) articles, though.

Enhancing the tree awareness of a relational DBMS S. Mayer (2004)

One can store an XML document (or any rooted tree) in a relational database by storing the preorder and postorder traversal rank (*i.e.*, the order of the opening, respectively closing, tags): this is the *XPath accelerator* coding – you can quickly get the set of preceding, following, ancestor, descendant nodes (those four operators (there are 13 in the XPath specification) are called *axes*).

An XPath expression starts with a (set of) node(s), applies one of the operators above to get another set of nodes, etc.

One can speed up the evaluation of an XPath expression by removing duplicates in those sets of nodes; by replacing those sets of nodes by a “minimum set” on which the next operator gives the same result (*pruning*); by scanning the table one (vertical or horizontal) slice at a time (*partitioning*) or even *skipping* some of those slices.

This speed up process is called the *staircase join*, because the sets of nodes, in the pre-, post-order plane, look like a staircase.

Fat tails and asymmetry in financial volatility models P. Verhoeven and M. McAleer (2003)

Even though GARCH models can account for fat tails, they rely on the unrealistic assumption that once the volatility is known, the returns are gaussian. Instead, one can use the *asymmetric T-distribution* – and the resulting GARCH parameter are more robust to outliers.

Return-based style analysis with time-varying exposures L. Swinkels and P.J. van der Sluis (2001)

The *Kalman filter* is an advantageous replacement for “rolling regressions”.

***Incorporating trading strategies
in the Black–Litterman framework***

F.J. Fabozzi et al. (Journal of trading, 2006)

Yet another introduction to the Black–Litterman framework.

***Portfolio risk forecasting*
G. Connor (2007)**

(These notes do not refer to the book, but to a series of lectures drawn from it.)

(1) The portfolio management setting

There is a difference between stock risk and portfolio risk: a portfolio is a linear combination of stocks, but it changes over time.

The book does not focus on *knightian uncertainty* – this notion is not defined, but it may be what you want if you seek a behavioral perspective.

The efficient market hypothesis suggests that you separate the returns as expected returns (that will disappear if you can forecast them and if people know how you do it) plus a zero-mean, pure risk return (that is there to stay).

Traditional portfolio optimization is myopic: it maximizes an investor’s utility over a single period. Multi-period, intertemporal approaches exist, but are not tackled in this book.

There are several notions of risk: *variance* (or rather its square root, the *standard deviation*, sometimes referred to as *volatility*), *value at risk* (VaR), *expected shortfall* (sometimes called *conditional value at risk* or CVaR) or even the full distribution of returns. The book will mainly focus on variance, the other quantities will appear as an overlay once the variance has been figured out.

Optimizers typically assume that the investor’s utility function has *constant absolute risk aversion* (CARA):

$$u(x) = -\exp(-\lambda x).$$

This leads to an optimization problem of the form

$$\begin{aligned} \text{Maximize} \quad & w'r - \lambda w'Vw \\ \text{such that} \quad & w'1 = 1 \end{aligned}$$

where λ is the absolute risk aversion of the investor, w are the portfolio weights, r are the expected stock returns and V is the variance matrix of the stock returns

You can solve that problem in various ways: either fix the portfolio returns and find the portfolio that minimizes the risk; or fix the risk and find the portfolio that maximizes the returns. Both computations can be carried out with Lagrange multipliers (the details were not given: check Campbell, Lo and MacKinlay,

The econometrics of financial markets) and actually yield the full efficient frontier.

Using active returns instead of total returns in the optimization yield inconsistent results – I do not know why.

The author stresses that *valuation models* and *risk models* are unrelated (indeed, they answer different questions) and, invoking Grinold and Kahn, suggested that the models even be orthogonal. (If you want to optimize a portfolio, I agree (but including the alpha components in the risk model is probably harmless, because you would then provide an alpha to the optimizer) and some people even suggest to *rotate* the risk model so that it be orthogonal to the valuation model used; but if you want to see if your portfolio corresponds to the bets you think you made, to quantify those bets, to check that there are no other unwanted bets, it might be better to include the valuation model in the risk model).

Risk is not additive across a portfolio, but if you want that property, you can use *marginal contribution to risk* (MCTR) instead. You can define the MCTR of a stock or of a “tilt” (a portfolio indicating the direction in which you might want to move your portfolio).

The CAPM is a decomposition of stocks returns into a market component and residuals (it does not claim that the residuals are gaussian iid: there is still some structure in them).

$$\text{returns} = \alpha + \beta \cdot \text{market returns} + \text{residual returns}.$$

This leads to a decomposition of the variance matrix of the returns into a market-related (scalar) variance matrix and a residual variance matrix.

$$\begin{aligned} V &= \text{market variance} \cdot \beta\beta' + \text{residual variance} \\ &= \sigma^2\beta\beta' + \Delta \end{aligned}$$

To get a risk model, you can simply impose some structure on that residual variance matrix – e.g., ask that it be diagonal.

Under this assumption, the market portfolio is mean-variance efficient.

The CAPM can be seen as a linear regression with a single predictive variable: you can have several, instead. This is the *Arbitrage Pricing Theory* (APT – this sometimes also stands for Advanced Portfolio Theory). This leads to the notion of factor model we are used to.

One can also build *factor mimicking portfolios*.

The book does mention the interpretation of the APT in terms of arbitrage – but not clearly.

The CAPM is a 1-period model: you can define an *Intertemporal CAPM* (ICAPM, Merton) instead: the factor exposures (betas) are not time-dependant.

The book also mentions the (admittedly useless) *Consumption CAPM* (CCAPM, Breeden), but not clearly.

(2) The structure of Portfolio Risk Forecasting Models

In a portfolio optimization problem, you have to provide expected returns and a variance matrix: these are not known precisely. One can decompose the utility to be maximized into a sum, the actual utility and a term resulting of the error on the expected returns and the variance matrix. When maximizing the utility, you are actually maximizing both terms – and it might be easier, faster to maximize the second one – this is the *error maximization problem*.

This chapter presents several (mainly bayesian) methods to mitigate that problem.

To tackle the effects of the estimation error on the expected returns, you can shrink those estimates towards more conservative ones:

- Jorion suggest a *Bayes-Stein prior*, i.e., the mean of the forecast returns as a prior (all the stocks have the same prior return), with variance equal to the sample variance of those forecasts
- *Black and Litterman* suggest using the alphas (expected returns) implied by the CAPM, i.e., those for which the market is efficient.
- Grinold suggests something complicated that is actually a special case of the Black-Litterman prior

You can take the same ideas to mitigate the effects of the error in the estimation of the variance matrix, by shrinking it to a more conservative prior, such as:

- a diagonal matrix
- a constant correlation matrix
- the variance matrix of a 1-factor (CAPM) model
- the variance matrix of a multi-factor (APT) model

Actually, it might be better and simpler to shrink it completely and to stay with the prior.

To mitigate the effects of estimation errors in a portfolio optimization, one can also add *position limits* – this can actually be reformulated as a *shrinkage estimator*.

Resampling to estimate the distribution of the alphas and the variance matrix can also help mitigate the problem: without position limits, it does not improve anything, but with position limits, we get a biased estimator that “might” be helpful. (The lecturer failed to mention that this bias can tell you to buy stock with a known null or even negative return.)

This chapter also stressed the difference between arithmetic returns and log-returns.

(1bis) Each lecture was followed by a discussion:

- The risk aversion parameter is not knowable
- Some people want the risk model to be orthogonal to the valuation model; other the former to contain the latter.
- The CAPM does not provide a decent risk model
- The market is not efficient: it is cap-weighted. (?)

(2bis) After the second lecture, there was a small discussion on the relative merits of

- Cross-sectional risk models (start with the factor ex-

posures use regression to get the factor returns: the residual variance, assumed to be diagonal, need not be so)

- Times series risk models (start with the factor returns and use regression to find the exposures: the residual variance is closer to being diagonal, but the number of parameters to estimate is larger, yielding higher estimation errors)
- Hybrid risk models (more about this in a forthcoming lecture)

(3) Industry and country risk

In short: there are very strong country effects, even in western Europe, and they are more important in smaller countries.

When people did not have stock-level data, they were using contry index returns and country weights.

The model is typically written

$$\text{returns} \sim 1 + \text{industry} + \text{pure_country}$$

where the intercept is the market returns, and the pure country returns are orthogonal (uncorrelated) to the industry returns, and estimated as a *random coefficient model* (or *mixed model*): this means that the coefficients of the regression (the exposures of the stocks to the country or industry effects) are considered as random (gaussian) variables. Most mixed models practitioners will focus on the mean of those random variables: we shall focus on their variance.

Estimating this covariance matrix can be tricky – and his confusing explanations do not help.

To account for the differing volatility of the stocks (heteroskedasticity), one can use a weighted regression: either using “feasable weighted least squares” (this sounds like iteratively reweighted least squares: first set the stock variances to the same value, fit the model, estimate those variances, use the to get another fit, iterate) or using root-capitalization weighted regression (but this introduces a bias, especially in concentrated markets).

You can test for the significance of the country (or industry) returns by performing monthly T- and F-tests and looking whether they are significant “overall”. (He does not seem to know anything about *multiple tests* and the corresponding *p*-value adjustments – or about *p*-values in the first place.)

This generalizes to industry and country weights, instead of boolean exposures, either by asking analysts to come up with those exposures for each stock, or by regressing the stock returns against the country and industry returns obtained on a first run using boolean exposures – but expect the results to be noisy.

This lecture was followed by a few comments by James Sefton (UBS):

- Is the cross-sectionnal method the best? It allows one to have time-varying betas – but here, they are set to 0 and 1... Iterative time series/cross-sectionnal approaches may be worth considering.

- Is the country effect really strong? Doesn't it stem from the inclusion of a few (small and) very different countries? There is a difference between the two questions:

Q1: Is a country index affected by its industry composition?

Q2: Is a given stock more sensitive its country or its industry?

Academics are interested in Q1, practitioners in Q2.

The lingering importance of countries may be linked to the very slow disappearance of the *home bias*.

(4) Statistical factor models

(4a) Structure of statistical factor models

The model underlying statistical factor models can be written as

$$\text{returns} = \alpha + Bf + \varepsilon$$

or, since we are interested in B (exposures, betas), f (factor returns) and the variance matrix of the returns,

$$\text{Variance}(\text{Returns}) = BB' + \text{Variance}(\varepsilon).$$

(With my usual notations: $V = EE' + \Delta$.)

Here, we want to estimate both B and f (in the previous chapter, we had B (boolean exposures, or exposures given by analysts) and we wanted the factor returns f ; in macro factor models, we have the factor returns f and we want the stock exposures B): there is some indeterminacy – one can multiply B by any invertible matrix.

Statistical models do not impose any prior judgement about which factors are relevant.

Statistical models do not require complicated data: the returns suffice.

One can define several more or less complicated models by imposing restrictions on the variance of epsilon:

- In a *noiseless factor model*, it is set to zero
- In a *scalar factor model*, it is scalar (diagonal with always the same value on the diagonal)
- In a *strict factor model*, it is diagonal
- In an *approximate factor model*, I do not know (it was not defined rigorously). Examples include block-diagonal matrices (say, one block for each subindustry (*)), or matrices whose non-diagonal elements decrease as they get far away from the diagonal – more generally, matrices with a lot of zeroes or a lot of small elements

(*) This suggests another way of finding clusters of stocks, to be interpreted as industries or sectors:

- Compute the variance-covariance matrix of the stock returns;
- Remove the effects of a few factors (either statistical factors or factors you can interpret);
- try to write the matrix as a block-diagonal one (hint: try correspondance analysis to reorder the rows/columns).

(4b) Estimation of variance matrices

The first step is to estimate the sample variance matrix. If there are no missing values, if there all the stocks have the same history length, it is easy to do. Otherwise, one can use the EM algorithm to fill in the missing early values.

The EM (Expectation-Maximization) goes as follows: from some data (X, Y) , with Y missing, you want to estimate some parameter θ . You can just iterate through the following two steps:

- E-step: compute the distribution (not the expected value) of the missing observations given the estimated parameters
- M-step: estimate the parameters given the distribution of the missing observations

One may want to reduce the amount of data (with fewer stocks, the estimates should be less noisy): this can be done by grouping the stocks (say, by replacing them by subindustry portfolios) or (this is mainly a historical curiosity) by sampling the universe.

On the other hand, one may want to increase the number of dates: this can be done by extending the period in the past (but things have changed since the 19th century) or by using higher frequency data: weekly or daily instead of monthly.

With “high” frequency data, you have to adjust for *stale prices*: simply replace the sample variance of the returns at time t by

$$\begin{aligned} &\text{Cov}(r_t, r_t) + \text{Cov}(r_t, r_{t-1}) + \text{Cov}(r_{t-1}, r_t) \\ &\quad + \text{Cov}(r_t, r_{t-2}) + \text{Cov}(r_{t-2}, r_t) \\ &\quad + \dots \end{aligned}$$

(If there are no stale prices, the added terms have a zero expectation.) Otherwise, the variance estimator is biased.

The *Solnik diversification curve* (of a universe of stocks at a given date) plots the sample variance of a random, equi-weighted portfolio as a function of the number of stocks in the portfolio. (Actually, its computation is straightforward and does not require any simulation.) It can be generalized to capital-weighted portfolios.

(4c) Small- n methods (few stocks)

Principal Component Analysis (PCA) can be used for the noiseless factor model (zero stock-specific variance – all but the first few eigenvalues are zero) or the scalar factor model (all the stocks have the same specific variance – all but the first few eigenvalues are equal).

For a strict factor model, $V = EE' + \Delta$, just consider $\Delta^{-1/2}V\Delta^{-1/2} = \Delta^{-1/2}EE'\Delta^{-1/2} + I$, which is a scalar model. Since you do not know Δ , start with an initial estimation of E (noiseless model), estimate Δ , then E , then Δ , then E , etc. – this is the *Joreskog algorithm*.

You can test the fit of the model with Likelihood ratio tests or with R^2 .

(4d) Large- n methods (many stocks, approximate factor model)

Asymptotic Principal Components (APC) (?), i.e., eigenvectors of $\frac{1}{n}R'R$ (we were previously considering $\frac{1}{n}RR'$), directly provide the factor returns.

As before, the EM-algorithm can be used to fill in the missing values.

Estimating the number of factors to retain by looking for a sharp drop in the eigenvalues is not reliable. You may have more success with the variance ratios or penalized likelihood (AIC, BIC).

(4e) Hybrid factor models

In a hybrid factor model, one first uses interpretable factors and then refines the residual variance matrix using statistical factors.

The discussion by Tim Wilding (EMA) mentioned the following points:

- The importance of Maximum Likelihood (ML) estimators
- *Independent Component Analysis* (ICA)
- The distinction between large- n and small- n methods is historical and irrelevant
- Even though G. Connor does not like statistical models, some factors are not knowable (Tim mentioned *behavioral factors*, Ruppert mentioned restructuring in Japan)

Jason MacQueen recalled that there is no “right” answer – only more or less useful ones.

Though the words *Random Matrix Theory* (RMT) were never uttered during the lectures, the “Asymptotic Principal Components” (APC) are extremely similar: RMT studies the distribution of all the eigen values of a random matrix while APC studies the distribution of the largest eigenvalue of a random matrix – since the idea is to discard the smallest eigenvalues, it is equivalent.

(5) The Macroeconomy and Security Market Returns

Macro-economic risk models are risk models whose factors are macro-economic time series, such as oil price, interest rate, unemployment, GDP, etc.)

Macro-economic factors need to be “detrended” before use. You can use *AR models*, Hansen-Rodrick models (?) or a *Kalman filter* to remove the *predictability bias* from the time series and get the innovations time series. For historical studies, you can also use an “AR” model that also includes the forward values of the time series – this introduces a *look-ahead bias*, but allows you to remove the predictability bias.

The sensitivity to those macroeconomic variables is very different in a bull or bear market: you can compute two sensitivities, a bull one and a bear one.

You can build mimicking portfolios, whose returns replicate those of a macro-economic factor – but they are often illiquid and have a high turnover.

Discussion (Ed Fishwick, Merrill Lynch):

- Macroeconomic factors are important (we speak of

them all the time), but they are not reliably used in quantitative risk models – they contain information, but it is hard to extract.

- Macroeconomic factors yield time-varying betas: jumps in prices (e.g., after earnings announcements) can lead to jumps in exposures.
- If you check the proportion of variance explained by macro-economic factors or by other risk model factors, you realize that they explain very little.
- Conclusion: macroeconomic factors contain relevant information, that cannot easily (linearly) be extracted.

Q&A: The useful variables depend on the investment horizon (in the short term, you have all the variables you are used to, in the long term, mainly the yield remains). For macro-economic factors, that horizon would be three months rather than one.

(6) Corporate Characteristics and Security Market Returns

In a statistical risk model, you have to estimate both the factor returns and the stock exposures to those factors. In a macro-economic (or cross-sectional) risk model, you know the factor returns and you compute the stock exposures (using cross-sectional regressions). In a *characteristic (or time-series) risk model*, you know the stock exposures and you compute the factor returns (using *Fama-MacBeth regression*).

Here are a few factors for equities: industry, country, size, value, momentum, volatility, liquidity, yield, index membership, ROE or ROE, leverage, currency sensitivity.

Here are a few factors for fixed income: ... (I might have missed a few ones), McCauley duration (?), twist factor (?), butterfly factor (?), convexity, options.

The factor returns can be computed by a regression or by *portfolio sorting* (or *quintile spreads*): the returns of a portfolio long the top quintile and short the bottom. Contrary to the mimicking portfolios of macro-economic factors, those portfolios are actually investible.

One can lower the number of parameters to be estimated in a risk model by requiring that the specific risk be a linear combination of the factors – you just have to estimate the coefficients of this linear combination, instead of each individual specific risk.

Discussion (Ian Paczek):

- The way you choose to tackle outliers (e.g., *wind-sorization*) has an effect on the estimation – that will end up biased or wrong.
- Question: can we get a daily risk model by tweaking a monthly one with an estimate of the daily volatility? (see chapter 13)
- When a physicist wants to use a signal at some frequency, it samples it at a higher frequency (the *Nyquist frequency*, explained somewhere in most computer music books, for instance <http://www-crcs.ucsd.edu/~msp/techniques/latest/>

book.pdf). Why don't we do the same thing in finance?

- Dummy (boolean) variables may not be a good idea: some companies work accross several industries.

Q&A:

- In a time-series model, the estimation errors in the exposures should diversify away when the number of stocks grows; in a cross-sectional model, they will stay.
- Momentum, as a risk factor, can be problematic, because of its time series properties (its computation involves a moving average) and it is only relevant for short-term forecasts.
- Risk model factors do have a predictive power in the long term, but it gets eaten by the increased kurtosis.

(7) Measuring and hedging foreign exchange risk

(7a) Foreign exchange risk

Currencies are very different from other classes of assets: they are symmetric (e.g., the returns of USD/EUR are the opposite of those of EUR/USD – this does not sound like a compelling argument: you can be long or short a stock), they tend to have fatter tails, and they are more volatile.

The speaker stresses the drawbacks of ratio returns versus *log-returns* – but since we eventually want portfolio returns, he decides to stick to ratio returns.

If you like formulas, here are a few (with log-returns, they are exact, with ratio-returns, they are approximations, valid as long as the returns remain small – this is called the approximate linear model):

$X^{\text{EUR/USD}}$: exchange rate

$F^{\text{EUR/USD}}$: forward exchange rate

r^{EUR} : returns in EUR of an investment in EUR

r^{USD} : corresponding return in USD

$r_{\text{Hedged}}^{\text{USD}}$: returns of the corresponding hedged investment

$r^{\text{EUR/USD}}$: currency returns

r_f : risk-free return

$$(1) r^{\text{USD}} = r^{\text{EUR}} + r^{\text{EUR/USD}}$$

$$(2) X/F = -r^{\text{EUR/USD}} + (r_f^{\text{USD}} - r_f^{\text{EUR}})$$

$$(3) r_{\text{Hedged}}^{\text{USD}} = r^{\text{EUR}} + (r_f^{\text{USD}} - r_f^{\text{EUR}})$$

$$(3') r_{\text{Hedged}}^{\text{USD}} - r_f^{\text{USD}} = r^{\text{EUR}} - r_f^{\text{EUR}}$$

$$(4) r^{\text{USD}} = (r^{\text{EUR}} + (r_f^{\text{USD}} - r_f^{\text{EUR}})) + (r^{\text{USD/EUR}} - (r_f^{\text{USD}} - r_f^{\text{EUR}}))$$

(The second term in the second equation is called the *carry*.) The third equation can be read as: you can eliminate currency risk, but you have to pay the *interest cost of hedging*; the fourth equation says that the excess returns are the same.

(7b) Currency hedging

To estimate the risk of a (non-hedged, international) portfolio, one can decompose it into hedged equities and currencies and compute the corresponding variance matrix: the diagonal blocks are the equity variance and the currency variance matrices, and the off-diagonal block is the covariance between those two asset classes.

The simplest currency hedging strategy consists in buying exactly the amount of currency forwards to balance the value of your equities: this is the *unit hedge*.

If the covariance matrix between equities and currencies is non-zero, the unit hedge is not the minimum variance hedging strategy – the problem is that this matrix is difficult to estimate, especially if the returns distributions have fat tails.

The use of ratio-returns instead of log-returns leads to an under-hedging bias, that can easily be corrected.

Those hedges are valid for a 1-year horizon, but can be divided by 10 on a longer (8-year) horizon.

(7c) Macroeconomic influences on currency returns

In the short term (up to 1 year), macro-economic variables cannot predict exchange rates, even if you cheat and use the future values of those variables – but in the longer horizon (4 years), it works better.

Exchange rates have some undesirable properties, such as the consequences of central banks interventions and currency crises (including *endogenous crises*, triggered by *herding behaviours* and leading to *contagion*).

Discussion (David Buckle, Merrill Lynch):

- He recalls a few formulas, and adds formula (4) to the list
- Benchmark providers do not hedge the currency returns but a forecast of it – but they fail to disclose how they compute this forecast.
- The minimum variance hedge is $h^* = -\beta$
- The mean-variance hedge is

$$\lambda^{-1} \frac{E[\text{currency}]}{\text{Var}[\text{currency}]} - \beta.$$

This is *very* sensitive to currency expectation: a variance of 0.2 standard deviations yields a 100% change in the hedge factor – what was believed to be hedged is not.

- The “compounding effect”, *i.e.*, the consequence of the choice of ratio-returns instead of log-returns, can be written as

$$E[\text{currency log-returns}] = 0$$

$$E[\text{currency returns}] = 12 \cdot \text{Var}[\text{currency returns}]$$

- (There was also a remark about *change of numeraire*, with a couple of formulas, but it was not very clear.)
- *Currency overlay* is a good idea
- Some people advocate *latent hedge*: the correlation of the returns of european stocks with the USD/EUR exchange rate should be -1 : if this is the case, you do not have to hedge international portfolios –

but, surprisingly, you should hedge domestic portfolios... If you believe that there is no correlation, you should hedge international portfolios, but not domestic ones. The truth is somewhere in between.

(8) Integrated risk models (“drill-down risk models”)

There might be some integration (*i.e.*, the distinctions between countries might be disappearing) for valuation models, but not for risk models, even in Europe. Some people saw an integration trend until 2000, but it was mainly due to the bubble. Here are several ways to build a world model, accounting for this segmentation.

(8a) Build local models and assume they are independent, *i.e.*, the global covariance matrix is block diagonal – this assumption is unrealistic.

(8b) Build local models and also build a global model – both models may be fine, but they fail to match...

(8c) To ensure that *drill-down consistency*, build a *second order factor model* with:

- a world market factor
- country factors
- global industry factors
- local industry factors, asked to sum up to zero
- global style factors
- local style factors, asked to sum up to zero

(8d) An *orthogonal double Procrustes* model is build as follows:

- compute the local models
- consider the space of global (non-parsimonious) models that specialize to the local model
- project the data onto that space to get the global model

Remark: I would personally add in some penalty to have a more parsimonious global model.

Discussion (Ely Klepfish, UBS):

- Are there links between the orthogonal double Procrustes problem and group representations of $SO(n)$?
- Do these models significantly differ from segmented model? (This question is worth asking: there are only theoretical results about this...)

Questions from the audience:

1. Can we do that with statistical models?
2. How does this compare with the Barra approach?
3. Even though there is a lot of segmentation, it is not enough to stick to country models.

(9) Dynamic volatilities and correlations

Traditional risk models try to estimate the variance matrix of the past returns, naively assuming that it will remain valid in the near future. Instead, one can model the dynamics of this variance matrix and forecast its next value.

For a single asset, one can use a *GARCH model* (the *news impact curve*, *i.e.*, the plot of the change in standard deviation versus the previous returns can help vi-

sualize the amount of information it brings), or its variants (but they assume that the frequency is fixed) or a *stochastic volatility* model (that does not make that assumption but whose estimation is only amenable to simulations).

The *realized variance*, *i.e.*, the integrated volatility on a time interval, can be seen as a model-free filter and could prove more useful than GARCH or SV models; even without intra-day data, the (under-used) *high-low range* is a good realized variance estimator.

The presentation of multivariate volatility models was confusing: in a nutshell, you start with a *VAR model* for the variance matrices (for n assets, there are n^4 parameters to estimate) and you try to reduce the number of parameters.

Examples of such models:

- Bollerslev’s *Constant Correlation* (CCor) model;
- Engle’s *Dynamic Conditional Correlation* (DDC) model;
- *Dynamic factor volatility models* (a factor risk model, with GARCH factors).

Downside correlation, *i.e.*, the increase in correlation when the market drops or simply becomes more volatile, is a problem.

The *dispersion* (*i.e.*, the cross-sectional standard deviation of the returns) changes with time.

Questions from the audience:

- High frequency data introduce fat tails, autocorrelation, covariances, etc.: is it really worth using them?
- Daily data (especially index data) are mainly used to *rescale* the variance matrix produced by risk models.
- Realized variance depends on the time interval – it should be proportional to it, but it is not.

Discussion (Dan diBartolomeo, Northfield):

- There are two kinds of innovations in the time series modeled by GARCH or SV models: *announcements* (we know the date beforehand but not the contents) and *shocks* (actual surprises). GARCH models cannot model announcements: they assume that the volatility has a trend, while the volatility will decrease before an announcement and increase after.
- Daily high-low ranges can be used to adjust variance matrices
- Beware when using dynamic factor models: the full variance matrix may be “strange” – you might need to add some constraints to get a sensible result.
- There are also non-parametric measures of correlation, such as *Kendall’s tau*.
- For more on dispersion, see the Northfield seminar, last month.

(10) Return Densities

(11) Liquidity and Credit Risk For credit risk, check *Credit risk: pricing, measurement, and management*, D. Duffie and K.J. Singleton, 2003.

(12) Long-short portfolios

(13) Long-horizon risk forecasting (five years and beyond)

The risk is supposed to be proportional to the square root of the time interval. The *variance ratio* statistic,

$$\frac{\text{Variance on } T \text{ periods}}{T \cdot \text{Variance on 1 period}}$$

measures the deviance from this rule (it is negative in the short term and the long term, positive in-between).

Returns are more volatile than company fundamentals: this *excess volatility* leads to *mean reversion*.

In the long term, variance is not really dynamic: we can forget chapter 9 (but see the *multiple component Engle-Ng model* (?)).

In the long term, we can no longer use ratio-returns: but luckily *log-returns* have almost the portfolio property.

Systems-based risk forecasting suggests to predict, not only the returns, but also the fundamental values (earnings, sales, GDP, etc.) that drive them, for instance with a VAR model – see *Strategic Asset Allocation*, Campbell and Viceira.

In the long term, *macro-economic factors* become important, differing *currencies* are no longer a source of risk but a hedge, and *active returns* disappears.

Risk forecasting cannot be separated from portfolio planning: rebalancing makes the portfolio risk non-linear and path-dependant.

Asset-liability management (?) is problematic, but beyond the book.

Discussion (David Miles, Morgan Stanley):

- Bonds are a good example of mean reversion: in the UK, they have been issued since the 17th century.
- Shouldn't we take care of rare events (world wars, 1929, 9/11, etc.)? See *Rare Events and the Equity Premium*, R.J. Barro, 2005. From a practical point of view, modeling rare events would be counterproductive.

(14) Portfolio risk forecast evaluation

This is important: you should assess your risk model before actually using it.

Fuzzy logic toolbox handbook Matlab

The idea of *Fuzzy Inference Systems* (FIS) is to replace boolean values by continuously varying values – think “probabilities”.

This can be generalized to boolean operators (**and** becomes **min**, **or** becomes **max**, **not(x)** becomes **1-x**) and *deduction rules* (**then** becomes **min**). Note that there might be incoherences in the rules – it is not a problem.

You can tune the fuzzy inference system by changing

the membership functions (or even the boolean operators).

As with bayesian methods, the result is a probability distribution, not a single number.

So far, it looks like bayesian networks, with a very simple, fixed network structure, and no learning capabilities.

Actually, you do not have to provide the set of rules: you can simply take your data, cluster it, and ask the computer to write a rule for each cluster. This is called *Adaptive Neuro-Fuzzy Inference Systems* (ANFIS)

This looks very similar to nearest-neighbour methods – and, if you impose some parsimony, it can be seen as a machine-learning analogue of Generalized Additive Models (GAM, **mgcv** package in R).

Fuzzy C-means clustering is a generalization of the *k*-means clustering algorithm where cluster membership is fuzzy – i.e., is a probability. (In R, it is implemented in the **fanny** or **cmeans** functions.)

For a different application of FIS, to control the level of detail in video games, as the number of objects in a scene increases, check *A fuzzy-control approach to managing scene complexity*, in *Game programming gems 6* (2006).

Maximum drawdowns of hedge funds with serial correlation B.T. Hayes (Journal of alternative investments, 2006)

Value at Risk (VaR) understates potential losses for less liquid and autocorrelated assets (e.g., hedge funds): instead, replace the returns by the *maximum drawdown at risk* (MDaR).

Its estimation is hampered by the short history of hedge funds, thereby calling for an *extreme value theory of drawdowns*.

This article suggests a (Markov chain, discrete) model (options people would call that a “tree” model) that accounts for autocorrelation but not for skewness and fat tails: the probability of an up or down movement depends on the direction of the previous movement.

This leads to explicit formulas – auto-regressive models only seem amenable to simulations.

The importance of being value F. Bourguignon and M. de Jong (Journal of portfolio management, 2005) J. Karvanen (2005)

Value (here, the price-to-book-value ratio (P/B)) can be decomposed into long-term P/B and current deviation from this long-term P/B; only the deviation leads to outperformance.

Price momentum and investor sentiment by sector

Y. Ishikawa (Nomura, 2006)

Stock-wise momentum does not work in Japan, but sector-wise momentum does.

Order imbalance and stock returns
H. Tamura and Y. Shimizu (Nomura, 2006)

The predictive power of the Order Flow Imbalance (OFI) is enhanced if you restrict yourself to the stocks with an auto-correlated OFI.

Estimation of quantile mixtures via L-moments and trimmed L-moments
J. Karvanen (2005)

The method of moments (equating sample moments to theoretical moments to estimate the parameters of your model) may look appealing (for instance, it allows you to dispense with gaussianity assumptions), but moments have a high variance (especially with small samples), are very sensitive to outliers – and need not even be defined for fat-tailed distributions. *L-moments* are defined using the *order statistics*: $X_{k:n}$ is the k th element of a sample of n observations from your distribution and

$$\begin{aligned}L_1 &= E[X_{1:1}] \\L_2 &= \frac{1}{2}E[X_{2:2} - X_{1:2}] \\L_3 &= \frac{1}{3}E[X_{3:3} - 2X_{2:3} + X_{1:3}] \\L_4 &= \frac{1}{4}E[X_{4:4} - 3X_{3:4} + 3X_{2:4} - X_{1:4}].\end{aligned}$$

Similarly, one can define *trimmed L-moments* (or *TL-moments*) by replacing $X_{k:n}$ by $X_{k:n+2t}$ for some value of t : they can be defined even for fat-tailed distributions with no mean. L-moments and LT-moments can be computed from the quantile function of the distribution.

The article advocates the use of *gaussian-polynomial* or *Cauchy-polynomial quantile mixture* distributions, *i.e.*, distributions whose quantile function is a sum of a gaussian or Cauchy quantile function and a polynomial: their parameters can easily be computed from the L- or LT-moments.

The article ends with a case study: skew Student, gaussian-polynomial and Cauchy polynomial quantile mixture distributions fit financial data (stock returns) equally well.

Biologically inspired algorithms for financial modelling
A. Brabazon and M. O’Neil (2005)

Multi-layer perceptrons (MLP) can be seen as parametric non-linear regressions, the number of layers indicating how deep the function calls can be nested. The functions you are combining can be linear (this is just linear regression), sigmoidal (S-shaped) or radial (bell-shaped).

Self-Organizing Maps (SOM, aka Kohonen maps) are a completely unrelated algorithm, sharing the same biological analogy (hence it is a *Neural Network* (NN)),

that can be used to find clusters in a data set – intuitively, it tries to cluster data while retaining information about the closeness of the observations by throwing a 2-dimensional net on the cloud of points.

Evolutionary algorithms (EA) take a population of potential solutions and mutate and merge them. *Genetic Algorithms* (GA) encode those potential solutions as binary strings; *Genetic Programming* (GP) encodes them as trees (a mathematical formula, a computer program, can be represented as a tree); *Grammatical Evolution* (GE) is a way of implementing a GP by encoding the tree as a binary string (the grammar is fixed and can include domain knowledge, but you can also evolve it: these are “GE by GE” or GE²). (If you are not familiar with BNF grammars, you can have a look at the Parse::RecDescent Perl module – there are similar modules for Python.)

One can combine Evolutionary Algorithms and Neural Nets, *e.g.*, to find which weights of the neural net are non-zero (MLP-GA) – this yields a more parsimonious model, less likely to overfit the data.

Differential Evolution (DE) is an evolutionary algorithm that explores the values of continuous parameters, where a new candidate x_{new} is obtained from three individuals x_i, x_j, x_k , as

$$x_{\text{new}} = x_i + \lambda(x_k - x_j).$$

Particle Swarm Optimization (PSO) is similar to DE and maintains a population of potential solutions, each being subject to three influences: inertia, attraction to the best solution found so far by the particle, attraction to the best solution found so far by the swarm.

Ant Colony Optimization (ACO) is a path-finding algorithm (but many combinatorial problems can be expressed in terms of the Traveling Salesman Problem (TSP), which is a path finding problem, so the algorithm is more general than it seems) that lets ants explore the various paths and mark the most promising path fragments by pheromones.

Ants can also be used as a classification tool (*Ant-Inspired Classification Algorithms*): put the objects to classify in 2-dimensional grid and have the ants move around and pick an object if there is no “similar” (for some distance) object nearby (say, in the last N cells visited) and drop it if there is (with some probability). This is very similar to SOM. Sadly, they conclude that ant classification does not outperform Linear Discriminant Analysis (LDA).

An *Artificial Immune System* (AIS) takes a model at random and rejects it if it does not work sufficiently well, until it reaches the desired number of models. This can be seen as a “Monte Carlo model selection” or as an EA with neither mutations nor mating, *i.e.*, with no evolution (some variants, such as the “clonal expansion and selection algorithms” do allow for mutations, though), followed by crude model averaging. AIS are often used in asymmetric situations, *e.g.*, to detect bad companies: healthy companies are homogeneous, but

companies can be sick in many ways – they are sometimes called *Negative Selection Algorithms* (NSA).

To cluster a large number of observations (think: points in a 2-dimensional space) you could divide the space into a grid and cluster the non-empty cells (in high dimensions, the grid would be sparse); *Artificial Immune Networks* (AiNet) proceed along a similar idea: take each observation as a center of a cluster, discard similar observations, link each remaining observation to its closest neighbours.

The second part of the book details ten case studies (equities, currencies, intraday trading, corporate failure prediction, bond classification) and gives potential inputs, fitness functions and post-processing ideas.

Here are a few performance measures, to be used as fitness functions:

- Mean Square Error (MSE, aka RMS or Root Mean Square);
- R^2 , AIC, BIC;
- Information Ratio (IR) or Sharpe ratio: returns / risk;
- Rank correlation (aka Information Coefficient or IC);
- Stirling ratio and its variants
return / drawdown
return – drawdown
return / drawdown \times win ratio
return / (1 + max(2%, drawdown)).

To examine a strategy, you can have a look at:

- the number of trades;
- the proportion of profitable trades;
- the average profit per trade;
- the average profit per profitable trade;
- the average loss per losing trade;
- the profit factor (ratio of the previous two averages);
- the p -value of a test for the positivity of the profits;
- descriptive statistics of the returns per trade;
- Sharpe ratio, Stirling ratio;
- maximum drawdown.

The model found by the algorithm should pass the *intuition test*: it should be understandable and meaningful. For neural networks, this might be difficult – but you can compute the *contribution* of an input variable (the sum of the absolute values of the weights between that input and the hidden layer, divided by the sum of the absolute values of all the weights to the hidden layer). Neural network with only a handful of non-zero weights (e.g., MLP-GA) are easier to interpret.

The book does not mention *bayesian networks*, that are very similar to neural nets, but much easier to interpret.

The *post-processing* of the models can include:

- *gating*, *i.e.*, inferring the “market regime” and using one of several models accordingly;
- *soft gating*, a market-regime-dependant mixture of models;
- stacking (averaging) of several models.

Here are some of the technical variables one may want to include:

- Moving Averages (MA);
- MACD (MA Convergence Divergence) oscillators (an oscillator is a mean-reverting signal, that suggests an action should be taken when it crosses its mean – here, when it crosses zero), *i.e.*, differences of MA with different window sizes;
- Momentum;
- *Trading range breakout*: take an action if the price move beyond its n -day maximum or minimum; *Bollinger bands* are a variant of this idea, with standard deviations instead of extrema;
- $\%K = (\text{close} - \text{low}) / (\text{high} - \text{low})$, where high and low are taken on an n -day window;
- $\%D$ is a moving average of $\%K$;
- *Relative Strength Indicator* (RSI): $R / (R + S)$ where R is the average of the positive returns and S the average of the losses (people usually write it as $100 - 100 / (1 + R/S)$);
- Volume (which is a measure of volatility and can be used as a volatility parameter in GARCH-like models);
- Ease of Movement (OEM) – their definition has a unit problem.

You can also add in market sentiment indicators:

- indices;
- number of advancing/declining securities;
- number of stocks reaching a new high/low;
- short-interest ratio (?);
- volume of options traded;
- put options/call options;
- VIX (a volatility index).

Clusterv tutorial **G. Valentini (2005)**

To assess the reliability of a clustering, this R package perturbs the initial data set by projecting it onto a smaller-dimensional subspace, while trying to preserve the distances – this is designed to work in very high dimension, typically a few thousands.

Efficient computation of the skyline cube **Y. Yuan (VLDB 2005)**

The *skyline cube* is the set of skylines for all the possible set of dimensions – since the number of subspaces for which we want the skyline is huge, this calls for algorithms that can iteratively build the skylines.

Sometimes, you are not interested in the whole cube, but only in the cells above a certain threshold: the corresponding sparse cube is called an *iceberg*.

Catching the best views of skyline: a semantic approach based on decisive subspaces **J. Pei et al. (VLDB 2005)**

Databases contain more and more complex data (typically, graphs) and one may want algorithms tailored to one's specific needs (hence the need to be able to program close to the data, in the DBMS).

This article focuses on the *skyline problem*, i.e., the query of objects that minimize two (or more) attributes (say, distance from the beach and price, if the database contains information about hotels) – this can be seen as a combinatorial *efficient frontier* problem, the problem of finding the set of *Pareto-optimal* objects, the set of *maximal objects* (this is a partial order relation).

The articles finds all the subspaces (sets of dimensions) for which a given object is on the skyline.

***Bridging the gap between OLAP and SQL* J.-P. Dittrich et al. (VLDB 2005)**

An OLAP system is an interface to a database that performs aggregation in as many dimensions as desired (in R, it would simply mean writing things like `tapply(x, list(a,b), sum)`): if the data is represented as a hypercube, the OLAP layer computes and displays all its margins. Things can get more complicated by the fact that the factors used to aggregate can be nested (e.g., sector, industry group, industry, subindustry; year, month, day) or may overlap (e.g., year, month, week, day; or, for each year, “compute the moving average over the past three years”) – some of this is in the SQL-99 standard, but they awkwardly try to force the cube margins into a 2-dimensional table and end up using NULLs with two different semantics.

Furthermore, the OLAP engine should represent extracts of the cube margins in a 2-dimensional format (a *pivot table* in Excel-speak) and the user should be able to interactively select dimensions to add (*drill down*) or remove (*roll up*) – ideally without having to ask the DBMS to recompute everything.

In spite of this rather clear presentation, the authors do not seem to understand what a cube is, they seem to be unable to think in more than two dimensions – a syndrome of excessive spreadsheet usage. As many, they do not seem to know that the relational model does provide user-defined types (called “relational domains”) and can thus accomodate “cubic” data – current SQL RDBMS do need to be reworked, though.

***Heuristique pour l'optimisation difficile, les algorithmes de colonies de fourmis* J. Dréo et al. (2003, Eyrolles)**

Ant colony algorithms are used to solve combinatorial problems that can be expressed as a path-finding problem in a graph – this is more general than it seems: many problems can be reformulated as a Traveling Salesman Problem (TSP). The idea is to ask the ants to find a path and to “reinforce” the components of the best paths found so far, by leaving *pheromones* on them. New ants will choose their path depending on those pheromones. The algorithm can be refined by taking into account the attractivity of the path com-

ponents, pheromone evaporation, by allowing the ants to disregard pheromones to increase diversification.

***An introduction to econophysics, Correlations and complexity in finance* R.N. Mantegna and H.E. Stanley (2000)**

From the central limit theorem, the Gaussian distribution can be seen as an “attractor” in the space of probability distribution functions; but its basin of attraction does not cover the whole space and it is not the only attractor: the other attractors are called *stable distributions* – this is the *generalized central limit theorem*.

Such stable distributions, e.g., the Cauchy distribution, often have no *characteristic scale*: the basin of attraction of the stable distributions contain the distributions with *power-law* tails.

You can refine the cartography of the space of probability distributions by considering the *infinitely divisible random processes*, i.e., distributions of random variables that can be written as sums of n iid random variables, for all n : Poisson, Gamma or stable distributions are infinitely divisible, while uniform distributions are not.

All those properties can be investigated through the Fourier transform.

The authors present and compare several definitions of returns and suggest to stick to *log-returns*.

You may want to study your data with respect to different *time scales*: physical time; physical time minus nights, week-ends and holidays; number of transactions; volume.

The autocorrelation function R (physicists would call that a *two-point function*, as opposed to *higher order statistics* or *n-point functions*) and the *power spectrum* S contain the same information and can be used to measure the memory of a time series. The process has *short memory* if $\int_0^\infty R$ is finite, e.g., if $S(f) \sim 1/f^2$ – one can then define an (ad hoc) *characteristic time scale*. It has *long memory* if that integral is infinite, e.g., if $S(f) \sim 1/f$ – this can be approximated by a process with many time scales. Note that this integral can be indefinite.

To that regard, financial time series exhibit an unusual behaviour: though the returns have a short memory, the volatility (roughly speaking, the squared returns) have a long memory – higher-order statistics are relevant.

One can try to account for the fat tails of returns by modeling them as stable distributions, T distributions, mixtures of gaussians, truncated stable distributions.

For small values, stock returns are well modeled by a stable (non-gaussian) distribution; for larger values, the tails are too thin for the stable distribution but too fat for a gaussian one.

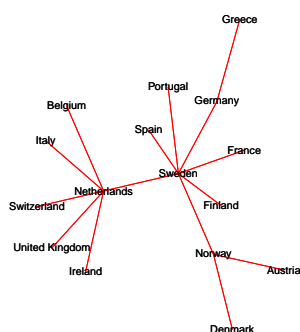
GARCH processes (with gaussian innovations) are fat-tailed (you can even compute the kurtosis, if you want), they fit stock returns much better than gaussian or stable distributions and they have non-trivial higher order statistics.

The book devotes a chapter to the similarities between stock prices and turbulence phenomena in fluid mechanics – only to conclude there are none.

The distribution of the correlation of stock returns changes over time and gets closer to zero. They mention *random matrix theory*, with neither details nor results.

You may want to check the cross-correlations (after a lag): there is some information, larger capitalizations tend to move ahead of smaller ones.

A correlation matrix can be interpreted in terms of distances which, in turn, can be represented as a Minimum Spanning Tree (MST – if you want to look cultivated, you can speak of *ultrametric distance*) or a dendrogram; you can do this on your whole universe or on the stocks in your portfolio. They fail to see that dendrograms are badly affected by fat tails and are far from stable.



The book ends with an introduction to options: first on an idealized market, then on a real one, that complicates the picture (discontinuous stock returns, unknown and changing volatility, discrete time, transaction costs, round lots, non-gaussian distributions): replicating portfolio and perfect hedging strategy do not exist.

Among the interesting plots:

- Probability distribution function with a vertical log-scale, to assess the fatness of the tails;
- Autocorrelation function with a log-scale on both axes;
- Spectral density with a log-scale on both axes;
- Probability distribution function of the stock returns over time (3-dimensional graph);
- Probability of returning to the origin after n steps, with a log-scale on both axes;
- Histogram of the correlation matrix of the stock returns, over time.

Quantitative stock selection in Japan and the United States: some past and current issues **J.B. Guerard (2006)**

The author computes Information Coefficients (ICs) for a dozen yearly, quarterly, monthly factors.

They perform a robust (biweight) “latent root regression”: robust to account for the outliers, “latent root” to account for multicollinearity (“latent root” seems to be a weird synonym of “eigenvalue” – I do not know how “latent root regression” relates to PCR (Principal Component Regression), PLS (Partial Least Squares) or ridge regression).

Bayesian clustering of many ARCH models **L. Bauwens and J.V.K. Rombouts**

Instead of fitting a GARCH model to each time series in you sample, you can try to fit your data as a mixture of GARCH models. As usual with mixtures, you can fit them with bayesian methods (MCMC: Monte Carlo Markov Chains).

This can be seen as a clustering method or as a poor man’s mixed GARCH model.

Multiple alpha sources and active management **E.H. Sorensen et al. (2004)**

This article builds on very unpedagogical foundations: Grinold and Kahn’s book.

They try to devise “optimal” (in the sense that they maximize the rank correlation with the forward returns – this is a form of robust regression, albeit not a very academic one) weights to combine two factors using their ICs – forgetting that the ICs are relevant for non-gaussian data while they assume theirs are gaussian.

In a later article, the same authors claim that this article claims (or even “proves”) that the optimal weights are

$$\mathbf{w} = (\text{Var IC})^{-1} \cdot \text{IC}.$$

Temporal data and the relational model **C.J. Date et al. (2003)**

The relational model can handle temporal data: just define an “interval” type and a few operators to handle them – optimizing the queries is a non-trivial and non-tackled *implementation* problem.

“Time point” types are granular: they come with “next” and “previous” operators, e.g., “date with daily granularity”, “date with monthly granularity”, “date with business day granularity”, etc.

PACK and UNPACK operators, that merge overlapping/abutting intervals and that split intervals into unions of 1-point intervals, are handy to write queries or constraints – e.g., primary key constraints.

Other operators can be readily generalized: unpack everything, apply the operator, pack the results. A few intricacies arise, because in higher dimensions, *i.e.*, if

you have several interval columns, the packed form is not unique.

A semi-temporal (or *current*) table is a table with a SINCE column.

A *historical* table is a table with a DURING column.

The book suggests to avoid a NOW special value (its semantic would be changing...) and to avoid using the end-of-time timestamp for open-ended intervals (indeed, that would mean putting incorrect values in the database): instead, use two sets of tables, one for historical data (with a DURING column) another for current data (with a SINCE column) – but updating the tables (since it may involve moving data from one set of table to another) or writing the constraints becomes inordinately complex.

Bitemporality is briefly mentioned: “The database is not the database. The log is the database. The database is merely an optimized access path to the most recent version of the log.”

The book uses the vocabulary of the relational model (“relation” instead of “contents of a table”, “relvar” instead of “table”, “attribute” instead of “column”, “tuple” instead of “row”) and the examples use the *Tutorial D* language (recalled in the introduction).

There is an *annotated* bibliography.

Needles, Haystacks and hidden factors **G. Miller (2006)**

In a risk model, statistical factors can be used to complement fundamental factors and help mitigate portfolio risk under-estimation.

(The article distinguishes between three kinds of factors: *fundamental factors*, whose exposures are estimated by hand and whose returns are computed from the exposures; *macroeconomic factors*, whose returns are known and whose exposures are computed from the returns; and *statistical factors*.)

Independant variable selection: application of independant component analysis to forecasting a stock index **A. Cichocki et al. (2005)**

The article generalizes PCR (Principal Component Regression) by replacing PCA (Principal Component Analysis) by ICA (Independant Component Analysis) – ICA is used for source separation in acoustics and to build some statistical risk models (perhaps APT).

The authors also add a bit of neural networks, because it is trendy – but they apparently fail to understand what ICA is...

Introduction to modern portfolio optimization **B. Scherer and D. Martin (2005)**

Chapter 1 introduces linear and quadratic optimization – but assumes you already know about scenario optimization or the Black–Litterman model.

Quadratic optimization, aka *mean-variance optimization* or *Markowitz optimization* or *modern portfolio theory* goes as follows: you have n (jointly) gaussian random variables $X = (X_1, \dots, X_n)$, you know their mean $\alpha = E[X]$, their variance matrix $V = \text{Var } X$ (that matrix is also called a *risk model*) and you want to find a linear combination $w_1X_1 + \dots + w_nX_n = \mathbf{w}'X$ (*i.e.*, a *portfolio*) with $w_1 + \dots + w_n = \mathbf{w}'\mathbf{1} = 1$ under the condition that $\text{Var } \mathbf{w}'X$ be below some threshold (the risk constraint the client asks you not to breach), *i.e.*,

$$\begin{aligned} &\text{Maximize } \mathbf{w}'\alpha \\ &\text{such that } \mathbf{w}'\mathbf{1} = 1 \\ &\quad \mathbf{w}'V\mathbf{w} \leq \text{threshold} \end{aligned}$$

There are variants where the risk constraint is turned into a *soft constraint*, depending on the *risk aversion* parameter λ .

$$\begin{aligned} &\text{Maximize } \mathbf{w}'\alpha - \frac{1}{2}\lambda\mathbf{w}'V\mathbf{w} \\ &\text{such that } \mathbf{w}'\mathbf{1} = 1 \end{aligned}$$

Caveats:

- If two assets are correlated, the optimizer will not manage to distinguish them and the weights will vary a lot (but the sum of their weights will be better defined): this is the same problem as multicollinearity in regression.
- The estimation of α and V can have an impact on the optimal portfolio (nowhere in the book do they use a risk model: always the sample variance matrix, or a “robust variance matrix”).

If you have enough data, you can dispense with the gaussian assumption and use a *scenario-based optimization*. For instance, let S be a matrix whose rows are realization of the random variable X ; to test for *arbitrage* situations, one could solve

$$\begin{aligned} &\text{Minimize } \mathbf{w}'\mathbf{1} \\ &\text{such that } S\mathbf{w} \geq 0 \end{aligned}$$

The first line is the money you invest (ideally, none) and the second is the requirement that the payoff be positive in *all* the realizations.

This first chapter also presents (or, rather, assumes that you already know) the Black–Litterman model and remarks that adding constraints to a quadratic optimization problem so that the solution looks better is equivalent to imposing a *view* – but a very strict one.

Chapter 2 presents the syntax of SIMPLE, the “language” used to describe the optimization problems (actually, it is plain S+, with a lot of placeholders, as in template-intensive numerical C++). It is a general optimizer: linear, quadratic, MIP (Mixed Integer Programming), convex, non-convex; the examples given range from Maximum Likelihood Estimators (MLE) to multistage stochastic programming to problems with CDS. The chapter also presents semi-quadratic, CRRA and other utility functions – but without a proper background in utility theory, this is not readily understandable.

Chapter 3 enumerates the extensions one can make to mean-variance optimization: constraints on the MCTR (Marginal Contribution to Risk), tracking a benchmark, tracking several benchmarks, using several risk models, using several risk measures, imposing lower bounds on weights, imposing bounds on the number of assets, including transaction costs, etc.

The more constraints you add, the weirder the efficient frontier and the more irregular the portfolio composition: a slight change in the acceptable risk threshold can yield drastic changes in the optimal portfolio composition – constraints actually chisel more edges on the simplex on the boundary of which the optimal portfolios lie...

Chapter 4 focuses on *resampled portfolios*, *i.e.*, portfolios optimal for resampled values of $\hat{\alpha}$ and \hat{V} ; similarly, a *resampled Sharpe portfolio* is the Sharpe portfolio for resampled values of $\hat{\alpha}$ and \hat{V} .

That notion provides a good argument against the long-only constraint: if you take many resampled Sharpe portfolios and average them, you get the true Sharpe portfolio; but with a long-only constraint, you do not – constrained optimal portfolios are biased...

Not bad enough? If one of your assets has zero expected returns (say, a lottery ticket), the long-only constraint will, on average, suggest that you buy it.

This chapter also suggests the use of the *Mahalanobis distance* (they almost clearly recall what it is)

$$d_{\text{Mahalanobis}}(\mathbf{w}_1, \mathbf{w}_2) = (\mathbf{w}_1 - \mathbf{w}_2)' V^{-1} (\mathbf{w}_1 - \mathbf{w}_2)$$

to compare portfolios. Some people also advocate the use of the *tracking error distance*,

$$d_{\text{TE}}(\mathbf{w}_1, \mathbf{w}_2) = (\mathbf{w}_1 - \mathbf{w}_2)' V (\mathbf{w}_1 - \mathbf{w}_2).$$

You can use them to design a statistical test to compare portfolios – but that test turns out not to have any power.

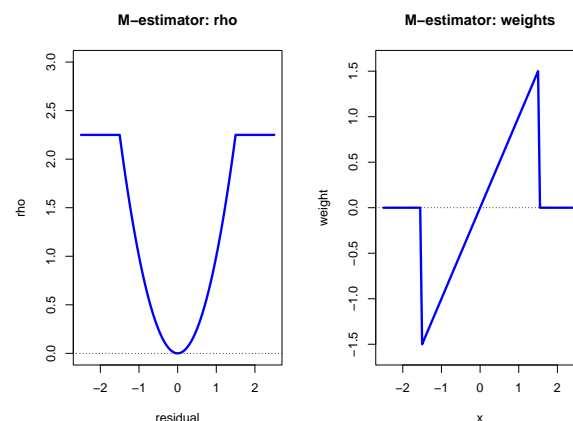
The chapter confusingly ends with a compendium of problems with resampled portfolios and/or the long-only constraint (mainly, they tend to invest in every stock) and advocates the use of robust methods.

Actually, resampling can prevent the adverse effects of the long-only constraint: construct resampled portfolios without the constraint and discard the stocks whose weight is not positive often enough – *i.e.*, is not significantly positive.

Resampling, aka *bootstrap*, can also give you an idea of the distribution of Sharpe or Sortino ratios – and *double bootstrap* (they try to explain what it is) can help lower the bias.

Chapter 5 addresses non-gaussianity, starting with *scenario optimization* and expected utility; they mention *copulas* but do not use them; finally they suggest replacing variance by MAD (Mean Absolute Deviation), semi-variance or CVaR (Conditional Value at Risk, aka Expected Shortfall or ES). The chapter closes on an application of scenario optimization and CVaR to CDOs.

Chapter 6, devoted to robust statistics, is the gist of the book. The most straightforward robust methods are the *trimmed* ones (trimmed mean, trimmed regression (LTS), estimation of the volatility of a time series where you discard the observations beyond 2.5σ , etc.). Where least squares methods minimize the sum of squared residuals, $\sum_i r_i^2$, an *M-estimator* minimizes $\sum_i \rho(r_i)$, for some function ρ . This is equivalent to a weighted regression.



The robust Mahalanobis distance between r_t (the returns of the stocks on a given day) and its historic (robust) mean can help spot hectic times.

When computing a historical variance, all the series need not have the same length: it is possible to compute a maximum-likelihood estimator of the variance matrix and the means that do not discard the information available (*Stambaugh method*); this method, in turn, can be robustified (they just give a recipe, with no theoretical foundation).

The book also presents “*robust portfolio optimization*”, *i.e.*, portfolio optimization with a robust variance matrix: if the robust efficient frontier and the classical one coincide, use the classical one; if they differ, investigate.

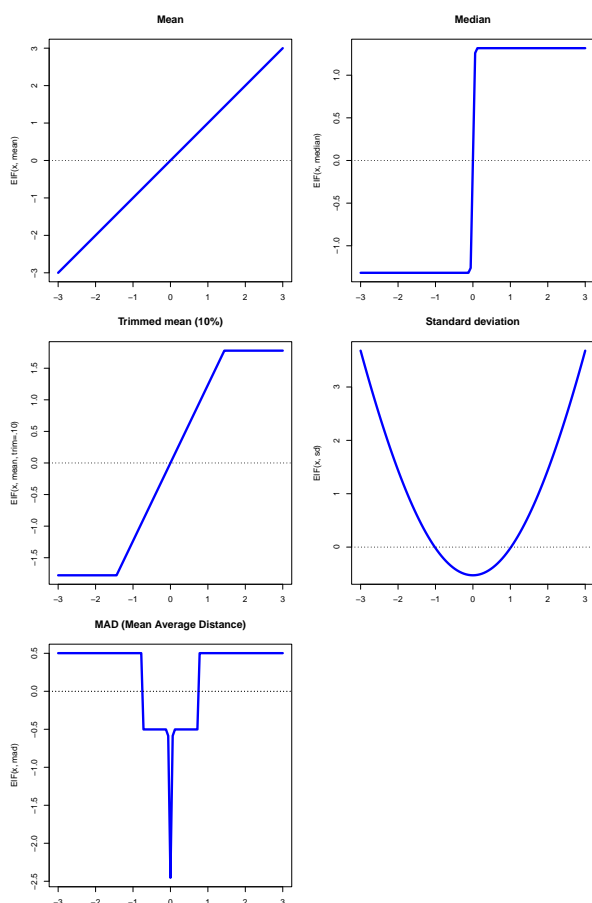
In these “robust portfolio optimizations”, we are concerned by the fat tails of the distribution of returns – and to solve the problem, we end up cutting those tails. To estimate the returns, *i.e.*, the center of the distribution, this is legitimate, but to compute the variance matrix, *i.e.*, the dispersion, this is dubious...

This chapter ends with an original section on *influence functions*. The *empirical influence function* or *finite-sample influence function* of a family of estimators $\hat{\theta}_n$ (e.g., $\hat{\theta}_n$ could be the mean of n numbers) is

$$\text{EIF}(x, \hat{\theta}, \mathbf{x}) = (n+1) \left(\hat{\theta}_{n+1}(x, \mathbf{x}) - \hat{\theta}_n(\mathbf{x}) \right),$$

i.e., it is the change in the estimator $\hat{\theta}_n(\mathbf{x})$ brought by a new observation x – the $n+1$ coefficient is a normalizing coefficient, ensuring that, under mild conditions, the large-sample limit exists.

```
## Error in huber(a)$s: $ operator not defined
for this S4 class
```



In a portfolio optimization setup, one can compute the influence of an asset (*i.e.*, the pair risk-return) on the risk of the optimal portfolio, its expected return, its Sharpe ratio, its weights – since there are two dimensions, the plot is actually a contour plot.

Chapter 7 is an unreadable introduction to bayesian statistics and Monte Carlo Markov Chain (MCMC) simulations. They insist on the importance of *Composition sampling* but fail to clearly explain that notion. The book ends on the Black-Litterman model, including a few interesting remarks on its generalization (use of a non-gaussian prior) and applications (to get an idea of the distribution of the Sharpe ratio) and the *Bayes-Stein estimator*, which they fail to clearly explain.

If you remember something from this book, it could be:

- Use robust estimators, *i.e.*, discard the most extreme observations – especially if your computations assume the data are gaussian;
- The influence function;
- The (robust) distance between today's return and the historical average to find "hectic times";
- The plot of the composition of a portfolio as a function of the risk (you might already be used to a similar plot with respect to time).

Quantitative strategy: global model portfolio
A. Tsai, Bear Stearns (2004)

They model stock returns as ARIMA processes, with

the parameters estimated for each stock, with the AIC, on a 30-month moving window; they claim that on a large, diversified universe (MSCI World), with a monthly rebalancing, with a 15% stop-loss, with a crude optimization (only consider the top 7% stock and minimize the risk, regardless of the returns) it works – with an 85% turnover.

If most of the stocks are AR(1), this is an exponential momentum factor whose length is stock-dependant. Their model captures short-term momentum but fails to spot longer-term reversal.

Separating the winners from the losers
A. Hartman et al., ABN-Amro (2003)

The article builds an adaptive model to predict future stock returns, as follows:

- Take 60 factors, compute their returns (how?), compute their robust correlation matrix, cluster them;
- In each group, select a factor with a high IC (Information Coefficient, *i.e.*, rank correlation with the forward returns) and a high mean(IC)/sd(IC);
- Model the monthly ICs of each factor as AR(1) processes, independently;
- These AR(1) forecasts of the ICs are too noisy: shrink them towards their historical mean (use the explained variance of the AR(1) model and the estimated variance on the estimator of the historical mean);
- Combine the factors using the following weights:

$$\mathbf{w} = (\text{Var IC})^{-1} \cdot \text{IC}.$$

Neural networks in business: techniques and applications for the operations researcher
K.A. Smith and J.N.D. Gupta (2000)

A review articles on the history of neural networks, the families of neural networks (multi-layer feed-forward with back-propagation learning rule, Hopfield networks (for quadratic programming), Kohonen maps) and their applications.

Surrogate time series
T. Schreiber and A. Schmitz (1999)

A more detailed (but less readable) article about surrogate time series, stressing what you can and cannot test with them – and how powerful or powerless those tests are:

- For the ARMA surrogate data, they allow for the series to be transformed in order to be gaussian;
- For the Fourier Transform-based surrogate, they use an iterative process to compensate for some of the artifacts of the naive approach.

The non-linearity tests they suggest rely on a non-linearity measure, such as:

- The third-order statistic, $E[(X_n - X_{n-\tau})^3]$, that measures the asymmetry of a time series under time reversal;

- the RMS (Root Mean Square) error of a locally constant (k nearest neighbour) predictor;
- Coarse-grained redundancies (?);
- Symbolic methods (?);
- False nearest neighbour (?);
- Unstable Periodic Orbits (UPO) (?);

The appendix of the article discusses an implementation of those ideas: TISEAN – you should also have a look at the tseriesChaos package, in R.

***Detecting nonlinearity in data
with long coherence times***
J. Theiler et al (1992)

Bootstrap is not as straightforward for time series as it is for other types of data. In this context, resampled time series are often called *surrogate data*. Here are a few ideas:

- Bootstrap as with other kinds of data – but this assumes that the observations are independent and destroys any causality information;
- Cut the time series in chunks and reshuffle them – this is not as bad, but it creates structural breaks in the series;
- Fit an ARMA model to your data, reshuffle or resample the residuals and reconstruct the ARMA series;
- Idem, but sample the new residuals from a gaussian (or whatever) distribution;
- Compute the Fourier transform of your series, keep the amplitude, replace the phase by random data and take the inverse Fourier transform.

The result, however, may be quite different from the initial time series, e.g., for the following reasons:

- The signal is periodic and there is not an integral number of periods in the window used for the Fourier transform;
- Changing the phase at the main frequency of the signal and/or at its harmonics can have strange effects;
- The time series is not stationary;
- The time series is non-linear (surrogate data are often used to test for non-linearity, but one should be aware of these and other artifacts).

***The S^2 tree: an index structure for
subsequence matching of spatial objects***
H. Wang and C.-S. Perng

To index sequences of points (“trails” as used, e.g., for subsequence matching in time series databases), one can combine ideas from sequence indexing (suffix trees) and spatial data indexing (R^* -trees), as follows: build an R^* -tree with the points in your trail or the MBR (Minimum Bounding Rectangles) covering them and add the links you would have in a suffix tree.

***Efficient time series subsequence matching
using duality in constructing windows***
Y.-S. Moon et al. (2000)

This article refines the previous: instead of sliding a

window on the data sequences, one can cut them into chunks; one then has to slide a window on the query sequence, instead of cutting it into chunks.

***Fast subsequence matching
in time-series databases***
C. Faloutsos et al.

To look for subsequences in a time series database, preprocess it as follows:

- move a sliding window (of the same length as the query series, or smaller) in the data series;
- For each position of the sliding window, compute a DTF (Discrete Fourier Transform) or any other distance-preserving transform;
- Retain the first k coordinates: you get a *trail* in a k -dimensional space;
- Since the trail contains too many points to store them all, cover the trail with rectangles (MBR, Minimum Bounding Rectangles);
- Store those rectangles in an R^* -tree: it is a data structure to store or index geometric objects that allows you to efficiently look for objects that intersect a given query object (it is used by some spatial databases).

***Forecasting transaction rates:
the autoregressive conditional duration model***
R.F. Engle and J.R. Russell (1994)

Autoregressive Conditional Duration (ACD) models are models of irregular time series, very similar to GARCH, with the variance replaced by the duration (*i.e.*, the time between two values) – so similar that one may use GARCH software to fit those. It can also be seen as a State Space Model (but a non-gaussian one: durations are waiting times). The model can be generalized to accommodate for discrete increments.

A portfolio diversification index
A.M. Rudin and J.S. Morgan
Journal of Portfolio Management, 2006

To check if the risks in your portfolio are well diversified, you can:

- Look at the variance matrix of the returns of the assets in your portfolio (but you need a good pair of eyes for this: the matrix is huge);
- Cluster the stocks in your portfolio and count the number of clusters (but this is somewhat arbitrary).

This article suggests to count the number of “independent factors” in your portfolio, as follows:

- Perform a Principal Component Analysis (PCA) on the variance matrix of the stock returns, to get those “independent factors”;
- Compute the weights w_k of each factor k in your portfolio;
- Count the number of factors, as follows:

$$\text{PDI} = 2 \sum_k k w_k - 1,$$

(if there is a single factor, *i.e.*, if $w = (1, 0, \dots)$, then $\text{PDI} = 1$; (if there are n factors and $w = (1/n, \dots, 1/n, 0, \dots)$, then $\text{PDI} = n$).

To check if your portfolio is diversified as it should, you can compare its PDI with that of random portfolios of the same size, on the same universe.

Stepwise portfolio construction using the *marginal PDI* does not significantly change the returns but lowers the volatility – when compared to random portfolio construction.

***Risk management for hedge funds:
introduction and overview***

A. W. Lo

VaR (Value at Risk) is not a good risk measure:

- It is not sub-additive, *i.e.*, the VaR of a more diversified portfolio need not be lower;
- It does not provide a decomposition of the risk into different risk sources;
- VaR is a static measure: it does not account for events (e.g., crashes), time-varying risks or dynamic trading strategies; it is computed unconditionally;
- VaR is difficult to estimate: if you choose Extreme Value Theory, you use very few data points; if you assume the distributions are gaussian, you are plainly wrong.

If you have to choose a hedge fund, do not be impressed by their positive returns: it does not mean that hedge fund returns tend to be positive, but simply that hedge funds with negative returns disappear (survivorship bias).

To show the problems of risk measurement, the article presents an unscrupulous but impressive trading strategy: short deep out-of-the-money put options.

To introduce events (crashes, “phase locking”), consider two risk models, one for normal days, another for event days. You can then use conditional measures of risk,

$$\frac{\text{Risk}[\text{portfolio} | \text{normal day}]}{\text{Risk}[\text{portfolio} | \text{event}]}$$

To account for non-linearities, do not compute sensitivities (β), but sensitivities (β_+ , β_-) to positive and negative values of the factor you are interested in (e.g., S&P 500) and test for their equality.

You can use Ljung & Box’s Q statistic (or rather its p -value) to measure the illiquidity of a fund.

***Time series analysis and prediction using
recurrent gated experts***
C. Gilde (1996)

Neural networks can be extended to accommodate time series, by reusing the previous value of a node, e.g., with a recurrent architecture.

This document adds that feedback capability to gated experts.

***Stock market pattern recognition
with neural networks***
D. Vengerov (1997)

The gated experts look at different time scales.

***Gated experts for classification
of financial time series***
D. Vengerov (1997)

To forecast future stock prices, use a Markov mixture of neural networks:

- The markovian structure recognizes the regime;
- The neural networks fit the data.

***Stock market prediction using
artificial neural networks***
B. Egeli, M. Ozturan, B. Badur

One can forecast future stock returns (or an emerging market index returns) with a

- MLP (MultiLayer Perceptron) – the neural net everyone knows;
- or a GFF (Generalized Feed Forward net) – in which the signal can jump one or several layers.

The input variables are: the previous value of the index (yes, a single value), the USD exchange rate, the day of the week. They forget to mention the number of neurons in the hidden layer – probably as many as input variables – and just focus on the number of hidden layers.

It works best with a single hidden layer and GFF networks.

21 nonlinear ways to beat the market
G.T. Albanis and R.A. Batchelor

Presentation of several Machine Learning (ML) algorithms,

- LDA (Linear Discriminant Analysis);
- PNN (Probabilistic Neural Network);
- RRI (Ripper Rule Induction algorithm);
- LVQ (Learning Vector Quantization);
- OC1 (Oblique Classifier)

and suggests to combine them with some voting scheme.

***Dynamic detection of change points
in long time series***
**N. Chopin,
Annals of the Institute of Statistical
Mathematics (2006)**

This article tries to account for the fact that GARCH models poorly fit long time series, suggesting that the model does not remain the same over time, by looking for abrupt model changes.

This can be done with MCMC simulations (remember that MCMC simulations may be used whenever you

have an algorithmic description of the process producing the data, with a reasonable number of parameters), but the algorithm will be slow (quadratic) and the convergence poor.

Instead, one can use a “particle filter”. The differences with MCMC are twofold:

- instead of generating several times series one after the other, one generates them at the same time;
- For each instant, we compute the likelihood of each time series, and their probability of survival to the next step is proportional to this probability (this should remind you of genetic algorithms).

The discrete nature of the abrupt model changes hinder the algorithm: the rest of the article explains how to circumvent this problem – the article could have been titled “Speeding up MCMC simulations in the presence of discrete variables”.

***Random portfolios
for evaluating trading strategies***
P. Burns (2006)

Traditional means of testing an investment strategy are very basic: rank correlations (called “information coefficients”) or return spreads (difference between the returns of the first and the last quintile of our signal or “alpha”).

But they forget half the story: how the strategy will actually be implemented. Instead of those crude measures, one can compare

- your strategy, *i.e.*, the transactions suggested by your alpha when fed to an optimizer;
- the strategy obtained by feeding a random signal (with the same statistical properties as your signal – do not forget the autocorrelation) to an optimizer (with the same constraints and the same risk model);
- a random investment strategy (with the same constraints: turnover, maximum and minimum weights, etc.).

Repeat this for hundreds of random signals (I would even suggest 10,000) to get an idea of how you signal fares. Feel free to change the initial portfolio (you might want to draw it “at random”, but this would have to be defined).

You may want to draw the following plots, either in back-tests or to monitor a strategy in real time:

1. cumulated returns of your portfolio over time, together with the fractiles of the cumulated returns of the random portfolios;
2. the p -value of the Stouffer test (?) over time;
3. final wealth of your strategy versus that of a random strategy, for the same initial portfolio.

You can also use those ideas to assess the influence of constraints, to compare risk models – or to compare optimizers.

***Copulas and coherence:
portfolio analysis in a non-normal world***

**K. Dowd
The Journal of Portfolio management (2005)**

They remark that correlation might not be a very good measure of the dependance between stock returns and suggest the use of copulas instead. As an illustration, they consider a two-stock portfolio: in their example, correlation-based methods vastly underestimate the risk. The problem, is that this is a simulated example, very far away from a gaussian setup: are dependances between real returns that extreme?

Another untackled question is the estimation of the copulas with a large number of stocks. In a correlation-based framework, this estimation is unreliable and we have to make further assumptions on the structure of the variance-covariance matrix, *e.g.*, that it only depends on a handful of factors: this is a risk model. One would therefore have to build our own copula-based risk model. This is an ambitious undertaking.

***Optimal trading strategies
and supply/demand dynamics***
A. Obizhaeva and J. Wang (2005)

Classical models to study trading strategies,

- We want to buy X shares of company RHAT
- We do it at times $0, \tau, 2\tau, \dots, N\tau$
- the price evolves as $P_{n+1} = P_n + \lambda x_n + u_n$ where $u_n \sim N(0, \sigma^2 \tau)$
- We want to minimize the transaction costs $E[\sum P_n x_n]$ or the risk-adjusted transaction costs $E[\sum P_n x_n] + \alpha \text{Var}[\sum P_n x_n]$,

whose solution is $x_i = X/(N+1)$, are overly simplified. For instance, in the continuous limit, they can no longer distinguish between the strategies: the transaction costs are strategy-independant.

The article provides a richer model, based on the Limit Order Book (LOB).

Portfolios from Sorts
R. Almgren and N. Chriss (2005)

This article develops a theoretical framework to define and compute an “optimal” portfolio from ordinal information (“this stock will outperform this one”, etc.), *e.g.*, from an ordering of the stocks. In a nutshell, they look for the portfolio that performs best in the worst situation that respects the ranking.

Surprisingly, this boils down to imposing a predefined distribution on the returns – but not a gaussian one.

This can easily be generalized to any partial order (*e.g.*, a total order in each sector but no information about one sector outperforming another) or any set of linear equations (*i.e.*, portfolios outperforming other portfolios).

This article is a simplification (a special case) of *Optimal Portfolios from Ordering Information* (2004), by

the same authors.

Barra factor returns and macro indicators

M. Furukawa and H. Tamura, Nomura (2005)

Application of the previous article to the Barra JPE3 factors:

- momentum (contrarian);
- value (especially when the yen is weak);
- interest-rate sensitivity (industrial production and topix decline, strong yen);
- leverage (rising long-term interest rate and topix);
- size (contrarian, rising topix).

The impact of the macroeconomic environment on global factor effectiveness

H. Tamura and K. Iro, Nomura (2002)

One expects factors to efficiently forecast future returns at some times and poorly at others, depending on the “macroeconomic environment”.

One can define the macroeconomic environment as follows:

- select a few macroeconomic variables (interest rates, stock market indices, exchange rates);
- apply a low-pass filter (there is an example on my web page; they seem to forget the boundary problems);
- check when the resulting signal is up or down.

One can define the returns of a factor as follows: use the (normalized) factor values as the active alpha in a mean-variance optimization; impose that the resulting portfolio be neutral wrt the other factors – actually, their description is rather vague: they could be using a Black-Litterman model.

One can then study the effects of the macroeconomic environment on the factor returns: this effect is important in the US and for the valuation factors (P/E, P/B, P/DPS, P/Sales). The most important macroeconomic factors are the long-term interest rate and the stock-market indices.

The results can be represented graphically in the returns×risk plane, with a point (mean risk and return) and two arrows (mean risk and return in the up and down phases) for each factor.

Out-of-core tensor approximation of multidimensional matrices of visual data

H. Wang et al. (Siggraph 2005)

A generalization of PCA that acts on matrices (or higher-dimensional arrays) instead of vectors, based on multilinear (aka tensor) algebra. This is used in computer graphics, to compress Bidirectional Texture Functions (BTF, *i.e.*, a texture is not given as a single image, but as a series of images that depend on the viewing and illumination directions). Applications in finance could revolve around volatility surfaces.

Genetic algorithms with collective sharing for robust optimization in financial applications

O.V. Pictet et al. (1996)

One can use genetic algorithms to design trading algorithms, as above: combine arithmetic operations, **sign**, **<**, **ifelse** and exponential moving average to produce a result in $\{-1, 0, +1\}$. They are careful to “normalize” the data and modify the division in order to avoid divisions by zero.

To stay away from non-representative peaks in the log-likelihood landscape, one can reduce the fitness in a region depending on the number of individuals in this region. This notion of “region” can be defined with gaussian kernels (but one has to choose the width of those kernels) or by applying the *k*-means clustering algorithm.

Genetic programming with syntactic restrictions applied to financial volatility forecasting

G. Zumbach et al. (2001)

Genetic programming can be used to perform model selection among a (huge) set of non-linear regression models, as follows:

- represent each model as a tree (the syntax tree of the corresponding formula);
- define a few mutation operators (node substitution, subtree mutation, root splicing, node insertion, node deletion);
- define a cross-over mechanism (given two trees, select a node in each and interchange the corresponding subtrees).

The article suggests:

- Do not use genetic programming to estimate the constants in the model, use more classical (and faster) algorithms;
- Use *typed* trees to preserve the symmetry in the function to be estimated;
- Fine-tune the mutation probabilities (they apply the genetic algorithm on simulated data with various values for the mutation and cross-over probabilities and select the best – with yet another genetic algorithm);
- Penalize the score of a given model by its “complexity” (which they define).

They then apply those ideas to FX data to forecast volatility (using simple arithmetic functions, absolute value, square and exponential moving averages).

The misbehaviour of markets

A Fractal View of Risk, Ruin and Reward

B. Mandelbrot (2004)

Besides providing a bird’s eye view of the history of finance, the author advocates three ideas.

1. Financial data exhibit fat tails, that follow a *power law*; this can be measured by the corresponding exponent, α .

2. Financial data exhibit long memory, or long-range dependance; this can be measured by the *Hurst exponent*, H . The interactions between those two effects can be measured by the *R/S statistic*.

3. Financial data look as though they were produced from a “shuffled” fractal process, with time distortions – but the book fails to give any detail as to the shape of those “time distortions”.

The geometry of crashes: a measure of the dynamics of stock market crises

T. Araújo and F. Louçã
arxiv:physics/0506137

During a crash, all the assets become correlated: this can be seen on the correlation matrix (estimated on a 3-week window) e.g., by comparing its first eigenvalues with those of “random” data. The article suggests a measure of that difference,

$$S = \sum_{i=1}^6 \frac{\lambda_i - \mu_i}{\mu_i}$$

where the λ_i are the actual eigenvalues and the μ_i those of “random” data.

There are a few problems, though:

- The article forgets to define “random”;
- Between the correlation matrix computation and the eigenvalues computation, the authors perform a (useless, if linear) MultiDimensional Scaling (MDS);
- It can only be used in retrospect, to conclude “this was not just an impression, there really was a crash”.

C++ design patterns and derivatives pricing
M. Joshi, CUP (2004)

This book has a very good but undeserved reputation (but again, its competitors might be even worse). It explains how to price derivatives, using Monte Carlo methods, in C++, with as many *design patterns* as possible.

Without *any* picture.

The author also explains how to circumvent some of the intricacies of C++ (as Meyer’s *Effective C++*, but more untidy) or some of its implementations (the *only* comment in the code is “...should be in namespace `std` but aren’t in `VCPP6`”) – at times, it even looks like an advocacy for interpreted languages (where you can write a Monte Carlo sampler in one line instead of one page) or fonctionnal languages (where a function can accept another function as argument).

A new method to estimate the noise in financial correlation matrices

T. Guhr and B. Hälber
arXiv:cond-math/0206577

Yet another correlation cleaning procedure: if the returns are the sum of sector-specific returns and

stock-specific returns, the correlation matrix is block-diagonal. The *power mapping* of the sample correlation matrix,

$$c_{ij} \leftarrow \text{sign}(c_{ij}) \cdot c_{ij}^q,$$

for a well-chosen (well, hand-chosen) value of q (this is merely a shrinkage of the correlation coefficients towards +1 or –1) provides a matrix that looks like the sample correlation matrix on a much larger time scale (two distinct peaks) and can thus be used instead.

Note that the authors used simulated data: I have tried it with real data (500 stocks, 300 days) – to no avail.

Financial applications of Random Matrix Theory: old laces and new pieces

M. Potters et al., arXiv:physics/0507111

Other correlation cleaning ideas:

- Shrink all the eigenvalues of the correlation matrix towards zero;
- Idem, but shrink more the small eigenvalues than the large ones.

Cluster analysis for portfolio optimization

V. Tola et al., arXiv:physics/0507006

The article presents three ways of filtering a correlation matrix to be used to build a portfolio:

- convert the correlation matrix into a distance matrix, perform an average linkage hierarchical clustering, measure the distance along the resulting tree, convert the (ultrametric) distance back to a correlation matrix;
- Idem with simple linkage;
- Random Matrix Theory (RMT, *i.e.*, PCA-based filtering).

The average linkage performs better.

Parametric portfolio policies: Exploiting characteristics in the cross-section of equity returns

M.W. Brandt et al. (2004)

Quantitative investors do not forecast the returns of each and every stock: instead, they devise a formula that will predict those returns from the characteristics (financial ratios, etc.) of the stocks; then they find the optimal portfolio for those returns. It is a two-step process.

Instead of predicting the stock returns, we can directly devise a formula to find the portfolio weights and tune this formula to optimize some “utility function”.

If you are used to classical portfolio construction, this looks a bit weird: the utility function you are used to requires the expected stock returns and the variance-covariance matrix of those returns. The article suggests a utility function (called the Constant Relative Risk Aversion (CRRA) function) that only depends on

the portfolio returns – a function of a single variable.

$$\text{Utility} = \frac{(1 + \text{return})^{1-\gamma}}{1 - \gamma}$$

You would expect the utility to depend on the portfolio expected return but also on its risk; you would expect the utility of a risky portfolio to be lower than that of a less risky one with the same expected return – but the formula only depends on the return – why?

The trick is that it indeed depends on the returns, not the expected return: we do not maximize

$$\text{utility}(E[\text{return}], \dots)$$

but

$$E[\text{utility}(\text{return}, \dots)].$$

The advantages over the classical paradigm are that:

- This is a *dynamic* utility function;
- The approach is not limited to the first two moments of the returns: it can also account for asymmetries or fat tails in return distributions.

A step-by-step guide to the Black–Litterman model T.M. Idzorek (2002)

Classical Mean-Variance optimization poses a few problems:

- The optimal portfolios tend to only contain a couple of stocks – they are not diversified;
- The optimal portfolios are corners (vertices) in a simplex: if we slightly change the inputs, we might jump to another corner – the portfolio is over-sensitive to the inputs;
- For the optimizer, the most important information often lies in the tiny differences between forecasts for similar assets, and it will amplify those differences, even if they are not significant, even if they are mere estimation errors.

The *implied returns* (or *implied alphas*) of the market portfolio are the expected returns for which the market portfolio is optimal: they are the expected returns α such that the market portfolio weights \mathbf{w} maximizes the utility $\alpha' \mathbf{w} - \frac{1}{2} \lambda \mathbf{w}' V \mathbf{w}$, where V is the variance-covariance matrix of the stock returns and λ the risk aversion, *i.e.*, the α such that $\mathbf{w} = (\lambda V)^{-1} \alpha$, *i.e.*, $\alpha = \lambda V \mathbf{w}$.

The *risk aversion* coefficient λ can be estimated as

$$\lambda \approx \frac{\text{risk premium (i.e., expected excess return)}}{\text{market excess return variance}}.$$

(Do not ask me why.)

The Black–Litterman model will combine those implied returns with a few *views* on the returns of a few stocks or portfolios. We need not provide expected returns for all the stocks (that would be error-prone) and we can simply provide the returns of portfolios instead

of individual stocks (it is easier) – forecasting the returns of a few portfolios instead of those of hundreds of stocks is sometimes called *mixed optimization*.

The Black–Litterman model relies on *bayesian statistics*: we have a prior belief (the returns will be gaussianly distributed, with mean the implied alphas and variance matrix V), we have some new information (or views: that the portfolios we can predict will have the returns ν we forecast, with a certain (diagonal) variance matrix Ω), and we combine all that.

Here is the classical set-up for bayesian statistics:

- We are interested in some parameter λ defining the distribution of a random variable, say $X \sim \text{Exp}(\lambda)$;
- We have some prior knowledge about λ : it is sampled from a known distribution, the *prior distribution*, say $\lambda \sim \text{LogN}(0, 10^6)$;
- We have some new information, usually a sample from the random variable X , say X_1, \dots, X_n ;
- We compute the distribution of the parameter λ given the new information X , *i.e.*, $\lambda|X$ – this is the *posterior distribution*.

This can be seen as an extension of Maximum Likelihood Estimation (MLE), whose prior contains no information at all, and whose result is the *mode* of the posterior distribution instead of the whole distribution itself.

If you want formulas for the probability distribution functions, this is the *Bayes formula*:

$$\begin{aligned} \text{Prior: } & f(\lambda) \\ \text{Posterior: } & f(\lambda|X) = \frac{f(X|\lambda)f(\lambda)}{\int f(X|\mu)f(\mu) d\mu} \end{aligned}$$

If you do not like probability distribution functions and prefer discrete probabilities, replace f by P :

$$P(\lambda|X) = \frac{P(X|\lambda)P(\lambda)}{\sum_{\mu} P(X|\mu)P(\mu)}.$$

But the Black–Litterman model does not follow those lines – it merely reuses the same ideas, “prior”, “new information” and “posterior”:

- Prior: the forward returns r should look as though they were sampled from a gaussian distribution, with mean the implied alpha α , with variance matrix τV , a scaled-down version of the variance matrix V of the stock returns provided by the risk model. Setting $\tau = 0$ would mean that we believe that the returns will exactly coincide with the implied alphas; some people suggest $\tau = 0.3$, others $\tau = 1$, others $\tau = 0.01$.
- New information: the returns Pr of the portfolios we can predict should look like they were sampled from a gaussian distribution with mean our view ν and variance Ω – a user-supplied matrix, usually diagonal to simplify things.

– Posterior: we try to somehow combine the prior and the new information. Algorithmically, one could proceed as follows:

- sample the portfolio returns Pr from their distribution $N(\nu, \Omega)$;
- sample the stock returns conditionally to the portfolio returns, *i.e.*, from $N(\alpha, \tau V) | Pr$;
- iterate and take the average.

Alternatively, you can *compute* the distribution of $N(\alpha, \tau V) | Pr$ (with the Bayes formula) (?)

$$-2 \log \text{Lik} = (x - \alpha)'(\tau V)^{-1}(x - \alpha) + (Px - \nu)' \Omega^{-1}(Px - \nu) + \text{constant}$$

and compute the Maximum Likelihood estimator. Equivalently, one can solve the following “GLS system”

$$\begin{aligned} r &= \alpha && \text{with variance } \tau V \\ Pr &= \nu && \text{with variance } \Omega. \end{aligned}$$

This yields

$$r = [(\tau V)^{-1} + P' \Omega^{-1} P]^{-1} [(\tau V)^{-1} \alpha + P' \Omega^{-1} \nu].$$

In this framework, the user has still too many parameters to specify: the views ν (this is the only one we want to provide the algorithm with), Ω (it says how confident we are in our view, but it is rather tricky to set) and τ . The article provides a simple and intuitive means of setting up those parameters. Well, actually, only Ω/τ appears in the formulas, so you can set $\tau = 1$ and only specify Ω .

Their recipe is as follows. For each view k (they consider only one view at a time):

- Compute the returns $r_{k,100\%}$ with $\Omega = 0$ (100% certainty);
- Compute the corresponding unconstrained weights $w_{k,100\%} = (\lambda V)^{-1} r_{k,100\%}$;
- Adjust the weights according to the confidence level C_k for the k th view ($C_k = 0$ for 0% confidence, 1 for 100%),

$$w_k = w_{\text{market}} + C_k(w_{k,100\%} - w_{\text{market}});$$

- Imagine that w_k comes from the Black–Litterman formula, for some value of ω_k : find the corresponding value of ω_k , in the Least Squares sense.

Then, put all the ω_k in the diagonal matrix Ω and apply the Black–Litterman formula.

Analysis of longitudinal data **P.J. Diggle et al. (2002)**

Chapter 1 contrasts *cross-sectionnal data* (a single observation for each subject) and *longitudinal data* (several observations for each subject, usually across time): different methods are needed to account for the lack of independence of the observations for a given subject and to avoid the *ecological fallacy*.

Here are three ways of dealing with longitudinal data:

- Marginal analysis: you look for the mean and the variance (at each point in time) over all the subjects;
- Random effect models: you have a model for each subject, $E[Y_{ij} | \beta_i] = x'_{ij}$, and the model parameters β_i are themselves iid random variables;
- Transition models: you have a model for $E[Y_{ij} | Y_{i,j-1}, Y_{i,j-2}, \dots]$.

I have not read the second chapter, *Design considerations*, that deals with power and sample size.

Chapter 3 suggests a few plots to explore longitudinal data:

- The variable to predict vs time, you can highlight some subjects (selected at random or not: *e.g.*, the lowest final value – or any other fractile); you may want to add a lowess curve;
- The variable to predict vs one of the predictive variables, again with a non-parametric curve estimation;
- The variable to predict vs its lags, as a pairs plot; the Auto-Correlation Function (ACF) – for irregularly-spaced and/or non-stationnary data, replace the ACF by a *variogram* – for binary data, replace the variogram by a *lorelogram* (LOR stands for Log-Odds Ratio).

Chapters 4 and 5, *Generalized Least Squares (GLS) and parametric models for the variance-covariance matrix*, present three correlation structures: the uniform correlation model, $\text{Cor}(X_i, X_j) = \rho$ if $i \neq j$ (equivalent to a subject-specific intercept), the exponential one (the equivalent of an AR(1) model for irregularly spaced data: $\rho(u) = e^{-\phi u}$) and the gaussian one ($\rho(u) = e^{-\phi u^2}$); an alternative approach is to perform a regression for each subject and then look at the distribution of the estimated coefficients – or, more rigorously, a random effects model.

GLS are simply weighted least squares, with weight matrix equal to the inverse of the variance matrix. One could use Maximum Likelihood (MLE) to estimate this variance matrix, but the results are biased: Restricted Maximum Likelihood (REML) gives better results (in short: with MLE, you estimate the regression coefficients β and the variance matrix V , but the estimators $\hat{\beta}$ and \hat{V} are not independent; with REML, you transform the problem into estimating β and another matrix H , so that $\hat{\beta}$ and \hat{H} be independent and that \hat{H} do not depend on β). For best results, estimate the variance matrix with as saturated a model as possible (*e.g.*, if no predictive variable is time-dependant, add a separate parameter for the mean response at each time with each treatment). The estimates of β , the confidence intervals and the tests remain valid even if the variance matrix is mis-specified.

When estimating the error, you can decompose it as a sum of three terms: the random (subject-specific) effects, the serial correlation, and the measurement (residual) error.

Chapter 6 explains how *Anova* (Analysis of Variance) provides a simple but unsatisfactory means of studying longitudinal data:

- With *cross-sectionnal anova*, *i.e.*, one anova for each time, you do not know how to combine the results (if they all agree and say there is something slightly significant, how do we combine them? how do we account for the correlation of the data in strengthening the significance of the result?) nor how to spot time-dependant effects (e.g., the slope);
- The idea of *derived variables* is to replace the observations for one subject by a single or a few numbers (say, the mean, the average rate of change, the parameters of a meaningful non-linear model) and perform an anova with this derived variable – but with several derived variables, you run into the same problems as before.
- *Repeated measures anova*, *i.e.*, a mixed model of the form $y \sim \text{treatment} + \text{treatment} : \text{time} + (\text{treatment}|\text{subject})$, completely forgets the time ordering of the data.

Chapter 7 presents the main models used to fit longitudinal data:

- *Marginal models*, *i.e.*, GLS: you specify the structure of the variance and of the correlations of the random variables;
- *Random effects models*;
- *Transition models*: you can model discrete data as a Markov chain whose transition probabilities follow logistic models.

Visual Explorations in Finance with Self-Organizing Maps **G. Deboeck and T. Kohonen Eds., 1998**

A *Self-Organizing Map* (SOM) or *Kohonen map* is a hybrid between the *k*-means algorithm and multidimensional scaling (it provides a clustering *and* a graphical representation of the proximity of the clusters), implemented via a “neural network” (a “non-linear, multi-layered, parallel regression technique” – if you already know about neural nets, bad luck, it has nothing to do with it: it stems from the same biological analogy, but it is a completely different species of neurons).

Intuitively, it takes a net and tries to cover the data with it.

The algorithm goes as follows: each node of the net corresponds to a cluster center; assign them to initial values (e.g., random values); for each observation, find the closest cluster center, drag it *and its neighbours* towards the observation (this is the difference with the *k*-means algorithm); iterate until convergence.

You will remark that SOM can handle missing values (however, for the observations with too many missing values, it might be wiser to remove them to compute the map and see afterwards where they belong).

You can also use SOM for forecasting purposes, in a *semi-supervised* way: use only the predictive variables to select the winning neuron, use all the variables to update the weights.

To speed up the building of the map and avoid insta-

bility problems with “large” (beyond 3×3) maps, you can start with a small map and later refine it – this is a *hierarchical SOM*.

A few ideas to display the map:

- Use a hexagonal grid with a neuron on every other tile; the other cells indicate if the nearby neurons are close (white) or not (black); this highlights frontiers between different regions of the map and/or areas of different densities;
- The *U-matrix* is similar, with a neuron on each tile and varying colours between the tiles;
- Colour the tiles according to the similarity of their neurons with their neighbours (aka *iso-contours*);
- Colour the tiles according to the number of observations they contain;
- The *Sammon projection* of the net, to see shriveled regions of the map;
- *Atlas* (or *component map*): colour the map according to one of the input variables (you get one picture for each variable, hence the name “atlas”); this can help interpret the regions of the map;
- Of course, it might be useful to display summary statistics for each cluster;
- Colour the map according to the *quantization error* (the distance between the observations and their neurons);
- Colour the map according to the *sensitivity* to changes in a component;
- The *synaptic weight map* indicates, for each neuron, which variable provides the greatest effect;
- Colour the tiles according to the curvature;
- You can imagine other ways of colouring the map: for instance, fit a circular (1-dimensional) SOM to the neurons and assign them rainbow colours;
- Parallel plot of the observations, with a different colour for each cluster.

Do not forget to estimate the quality of the map:

- its quantization error;
- its stability (try again with resampled data, without the outliers, with different parameters, etc.)

SOM are not only nice plots to look at:

- You can display the evolution of a subject (company) on the map (as the maps are not stable, you should reuse the same map);
- Given a new observation, you can highlight the closest neuron – or neurons;
- You can use a SOM for asset allocation: either invest in a single cluster, or invest equally in all the clusters, or assign weights to the clusters according to some other variables.

Some advice:

- Choose a rectangular hexagonal map: square maps are less stable; funny shapes (torus, Klein bottle) are only relevant if you *know* the data has this structure;
- Choose the number of neurons: either slightly less than the number of observations, if you want a compact map, or up to ten times the number of obser-

- ventions, if you want a more detailed map;
- Carefully choose the variables; you might want to discard some of them or even perform a Principal Component Analysis (PCA) and retain the first dimensions;
 - Carefully transform the variables, e.g.:
 - apply a logarithm or sigmoidal transform;
 - normalize them: $x \mapsto \frac{x - \mu}{\sigma}$;
 - range-normalize them: $x \mapsto \frac{x - \text{Min}}{\text{range}}$;
 - histogram-normalize them (this is a bad idea: it erases the clusters);
 - Transform unordered qualitative variables into binary variables;
 - Initialize the algorithm with PCA (only start with random data when you want to show that the map can self-organize – it works, but it is not optimal).

The book does not tackle variants of SOM:

- DEC (Dynamically Expanding Method);
- LSM (Learning Subspace Method);
- ASSOM (Adaptive Subspace SOM);
- FASSOM (Feedback-controlled ASSOM);
- Supervised SOM;
- LVQ-SOM (Linear Vector Quantization SOM).

The book provides examples in the following domains: Mutual funds, Firm failures, Russian banks, Emerging countries, Stocks, Real estate, Consumer preferences, Texts. You need not buy it: most of it is available on <http://www.dokus.com/PDF-files/>.

In R, have a look at `som` in package `som`, `SOM` in package `class` and `shardsplot` in package `klaR`.

Estimating the number of data clusters via the gap statistic

T. Hastie, R. Tibshirani and G. Walther (2000)

One can estimate the quality of a clustering with the *within cluster dispersion*, defined as the average distance between two points in the same cluster (weighted so that all the clusters have the same prevalence),

$$W = \log \sum_{r \in \text{Clusters}} \frac{1}{2|r|} \sum_{a,b \in r} d(a,b).$$

One can compare this number with that obtained from random data (uniformly distributed in a box aligned with the principal components).

The following algorithm suggests the number of clusters:

- Perform the clustering assuming $k = 1, 2, 3, \dots$ clusters and compute the within cluster dispersion W_k ;
- Do the same thing with random data and compute the average dispersion \bar{W}_k and its standard deviation s_k ;
- Define the gap as $\text{Gap}_k = \bar{W}_k - W_k$;
- Retain the smallest k such that

$$\text{Gap}_k \geq \text{Gap}_{k+1} - s_{k+1}.$$

Depending on the situation, the Gap statistic performs similarly or better than the Silhouette method.

Non parametric maximum likelihood estimation of features in spatial point processes using Voronoi tessalations

D. Allard

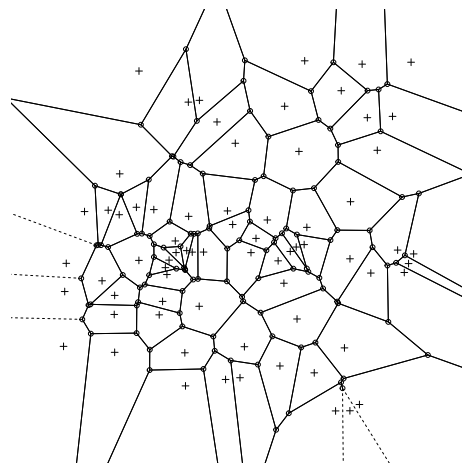
To denoise a spatial point process, one can model it as a mixture of two uniform variables, one on the whole space (the noise), one on a subset A of it. The partial log-likelihood (“partial” means that we have cheated: we need one more parameter, the probability of being in A , but we replace it by its maximum likelihood estimator) is

$$-n \log n + M \log \frac{M}{a} + (n - M) \log \frac{n - M}{1 - a}$$

where

- n is the total number of points;
- M is the number of points in A ;
- a is the volume of A .

The subset A is sought as a union of tiles from the *Voronoi tessellation* – actually, the union of the m smallest tiles, where the integer m is to be determined.



Nearest neighbor clutter removal for estimating features in spatial point processes

S. Byers and A.E. Raftery

To remove noise from a cloud of points:

- For each point, compute the distance to its k th nearest neighbour (the histogram of those distances should be bimodal);
- Model those points as a mixture of two Poisson processes, with different densities; estimate the model parameters with the EM algorithm (the model does not assume any shape);
- Label the points as “noise” or “not noise”.

The performance is close to that of the `mclust` algorithm, which is more computationnally intensive, and it works surprisingly well in high dimensions.

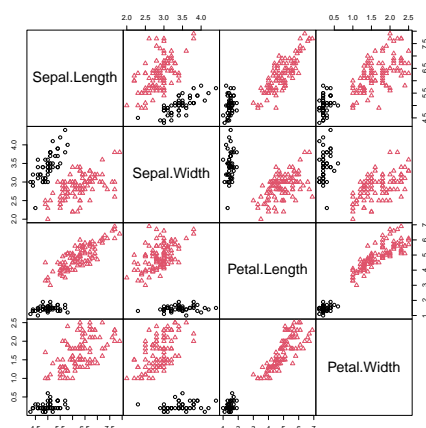
The idea can also be used to remove outliers.

In R, this is implemented as the `NNclean` function, in the `prabclus` package.

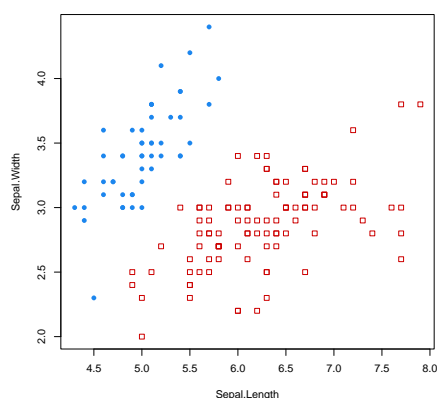
MCLUST: Software for model-based clustering, density estimation and discriminant analysis
C. Fraley and A.E. Raftery (2003)

Documentation of the `mclust` R package, that implements the EM model-based clustering algorithm – and its extensions to discriminant analysis and density estimation.

```
library(mclust)
r <- clustBIC(x)
plot(r) # BIC
s <- summary(r,x)
pairs(x, col=s$classification)
```



```
coordProj(x,
  z=s$z,          # Probabilities
  mu=s$mu,        # Cluster centers
  sigma=s$sigma,  # Cluster shapes
  what="classification")
```



Model-based clustering discriminant analysis and density estimation
C. Fraley and A.E. Raftery

Journal of the American Mathematical Association (2002)

The EM clustering algorithm can also be used in *density estimation* and *discriminant analysis* (the article has a lot of figures).

For high-dimensional data, new problems occur: the number of parameters (in the variance-covariance matrices) to estimate soars and calls for dimension reduction techniques (Principal Component Analysis (which may obscure the situation), Principal Curves, wavelets, Multidimensional Scaling (MDS)) or for extensions of the algorithms that directly deal with distance matrices.

Chameleon: a hierarchical clustering algorithm using dynamic modeling
G. Karypis et al.

Chameleon is a *graph-theoretic* clustering algorithm that takes into account the fact that clusters can have different densities and distinguishes clusters touching by a single point or a whole facet:

- replace the data by its k -nearest neighbour (sparse) graph, whose edges are weighted by the similarity of the vertices;
- find initial subclusters with the *edge-cut* algorithm;
- Merge the clusters whose Relative Interconnectivity (RI) and Relative Closeness (RC) are above a (user-specified) threshold, or according to the product $RI \cdot RC^\alpha$ (for a user-specified α).

The relative interconnectivity or closeness is defined from the absolute and internal interconnectivity (AI, II) and closeness (AC, IC), as follows.

$$AI(A, B) = \sum_{\substack{a \in A \\ b \in B \\ (a,b) \in \text{Graph}}} w(a, b)$$

$$II(A) = \min_{\substack{C \text{ is a} \\ \text{cut of } A}} \sum_{\substack{a,b \in A \\ (a,b) \in \text{Graph}}} w(a, b)$$

$$RI(A, B) = \frac{AI(A, B)}{\frac{1}{2}(II(A) + II(B))}$$

$$AC(A, B) = \text{Mean}_{\substack{a \in A \\ b \in B \\ (a,b) \in \text{Graph}}} w(a, b)$$

$$IC(A) = \text{Mean}_{\substack{a,b \in A \\ (a,b) \in \text{Graph}}} w(a, b)$$

$$RC(A, B) = \frac{|A \cup B| \cdot AC(A, B)}{|A| \cdot IC(A) + |B| \cdot IC(B)}$$

How many clusters?
Which clustering method?
Answers via model-based cluster analysis
C. Fraley and A.E. Raftery

The k -means clustering algorithm assumes that the clusters are symmetric and have the same size – hierarchical clustering algorithms similarly misbehave when confronted with clusters of varying shape, orientation

or size. The EM (Expectation-Maximization) algorithm finds the clusters by modeling the data as a mixture of gaussians, with potentially different variance-covariance matrices. The BIC (Bayesian Information Criterion) can help you choose the number of clusters and the variance-covariance matrix.

You can extend the algorithm to deal with noise:

- estimate the noise, by denoising the data (nearest neighbour method, Voronoi tessellations method);
- estimate the clusters, by a hierarchical classification of the denoised data;
- use the EM algorithm to fit the data as a mixture of gaussians (clusters) and a Poisson process (for the noise), initializing it with the previous two steps.

***CURE: an efficient clustering algorithm
for large databases***
S. Guha et al.

The *Birch* algorithm clusters large databases as follows: first replace dense regions by points (and discard less dense regions), then cluster those points.

The *CURE* (Clustering Using REpresentatives) algorithm is a variant of the *k*-means algorithm that does not use a single centroid for each cluster but *several* representatives: this allows for non-spherical clusters. You can adapt it to large databases by running the algorithm on a (random) sample of the data.

The article also briefly mentions other clustering algorithms for large databases: Clarans, R*-tree, DbScan.

***Determining the number of clusters/segments
in hierarchical clustering/segmentation
algorithms***
S. Salvador and P. Chan

There are four main categories of *clustering algorithms*:

- partitionning (aka *iterative relocation*, e.g., *k*-means)
- hierarchical (bottom-up or top-down);
- density-based (e.g., EM (Expectation Maximization));
- grid-based (e.g., SOM (Self-Organizing Map, aka Kohonen maps)).

Segmentation algorithms are related to clustering algorithms: they try to approximate a time series as a piece-wise linear function (PLR, Piece-wise Linear Representation); the main categories of clustering algorithms are:

- sliding window (grow a segment until the error exceeds a threshold, then start a new one);
- top-down (recursively split the entire series until the chunks can be approximated by segments);
- bottom-up (start with length-2 segments, merge the closest pair, iterate).

The number of clusters (resp. segments) can be guessed by:

- cross-validation;

- penalized likelihood:
 - MML (Minimul Message Length);
 - MDL (Minimum Description Length);
 - BIC (Bayes Information Criterion), AIC (Akaike Information Criterion);
 - SIC (Subspace Information Criterion);
- permutation tests (for segmentation: shuffle the time series);
- resampling;
- finding the maximum of a curve representing the quality of the cluster:
 - gap statistic;
 - prediction strength;
- finding the *knee* of the error curve.

The *L-method* finds the knee of an error curve by fitting it as a 2-segment broken line (to get more stable results, take a subset of the data so that there be as many points on each side th the knee; if the curve (or the time series) is too noisy, first smooth it by a moving maximum over a length 2 window).

Note that this does not work of there are only 1 or 2 clusters.

***Algorithms of maximum likelihood data
clustering with applications***
L. Giada and M. Marsili
arXiv:cond-mat/0204202

To cluster a cloud of points, the authors consider the following model: take n points at random, take k cluster centers at random, assign each point to a cluster at random, drag each point towards its cluster center. The likelihood is

$$\sum_{s : n_s > 1} \log \frac{n_s}{c_s} + (n_s - 1) \log \frac{n_s^2 - n_s}{n_s^2 - c_s}$$

where

- s is the cluster number;
- n_s is the number of points in cluster s ;
- $c_s = \sum_{i,j \in s} \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|}$.

Contrary to most methods, you do not have to specify the number of clusters. You can try to find the configuration with the best likelihood by the usual means (steepest descent, simulated annealing), which is quite time-consuming. The article suggests a faster (less precise) hierarchical algorithm: start with N 1-element clusters, merge two clusters so as to maximize the likelihood, iterate until there is a single N -element cluster, retain the clustering with the best likelihood.

The hierarchical clusterings they get in their two examples are very different: with financial data, single stocks are repeatedly added to already formed clusters, while with biological data, larger clusters are merged.

***Data clustering and noise undressing
of correlation matrices***
M. Marsili, arXiv:cond-mat/0003241

A correlation matrix estimated on a sample (and not on the whole population or from the probability distribution of the random variables) is “dressed in noise”, *i.e.*, not equal to the theoretical one. To see if there is something behind this noise, one can compare the spectrum (*i.e.*, the eigenvalues, sorted from the largest to the smallest) of the correlation matrix with that of the correlation matrix of a sample of uncorrelated gaussian random variables (this is called Random Matrix Theory (RMT) – if your distributions are not gaussian, just resample each vector independently) or that coming from a given model. The proposed model is hidden behind an overly specialized physics’s vocabulary: take n points at random, take k cluster centers at random, assign each point to a cluster at random, drag each point towards its cluster center.

*Noise dressing
of financial correlation matrices*
L. Laloux et al.
arXiv:cond-mat/9810255

*Marginal Contribution to Risk
for GICS industries in Aegis 3.4*
Barra (2004)

To compute and balance the risk of a portfolio of stocks, we use the variance-covariance matrix of the stock returns. As this matrix is huge, we do not estimate each of its coefficients: instead, we assume that the covariances are due to several “factors”; we estimate the variance-covariance matrix of those factors, we estimate the exposure of each stock to those factors (this means: we perform a regression of the returns of stock i against the “factor returns”, the exposures are the coefficients estimated by this regression), we estimate the residual variance of each stock returns and we put everything in a nice formula,

$$V = e'v e + \Delta,$$

where V is the variance matrix of the stock returns, e is the exposures matrix, v is the variance matrix of the factor returns, Δ is the (diagonal) matrix of the residual (aka stock-specific, systematic) variance.

If \mathbf{w} are the weights of a portfolio (as a fraction of the market value of the portfolio), then the risk (aka volatility, standard deviation) of this portfolio is

$$\text{Risk}(\mathbf{w}) = \sqrt{\mathbf{w}' V \mathbf{w}}.$$

The Marginal Contribution To Risk (MCTR) of a stock i is

$$\text{MCTR}_i = \frac{\partial \text{Risk}}{\partial w_i} = \frac{(V \mathbf{w})_i}{\text{Risk}(\mathbf{w})}.$$

Similarly, the contribution of a sector (seen as a portfolio \mathbf{s} to the risk of a portfolio \mathbf{w} is the directionnal

derivative

$$\begin{aligned} \text{MCTR}(\mathbf{w}, \mathbf{s}) &= \lim_{\varepsilon \rightarrow 0} \frac{\text{Risk}(\mathbf{w} + \varepsilon \mathbf{s}) - \text{Risk}(\mathbf{w})}{\varepsilon} \\ &= \frac{\mathbf{s}' V \mathbf{w}}{\text{Risk}(\mathbf{w})}. \end{aligned}$$

(You can do the computations yourself: this stems from the Taylor expansion of $\text{Risk}(\mathbf{w} + \varepsilon \mathbf{s})$.)

But how do we turn a sector into a portfolio? We could take each stock in this sector, with a weight equal to one; we could also weigh them by the market capitalization. Alternatively, if we do not have a single sector but a *partition* of the universe into several sectors, we can try to build a portfolio with unit exposure to the sector we are interested in and zero exposure to the others. This can be done as follows. Let f be the matrix of exposures to your sectors (one row per stock, one column per sector, 1 if the stock is in the sector, 0 otherwise) and δ_j the desired exposure for the sector portfolio (1 in position j and 0 elsewhere). We want a row vector \mathbf{s}_j such that $\mathbf{s}_j f = \delta_j$: it is not unique, but the pseudo inverse of f gives one: $\mathbf{s}_j = (f' f)^{-1} f' \delta_j$.

Actually, these are *not* the formulas mentionned in the article: for a reason I do not understand, they first compute the contribution of their factors to the portfolio risk and then “translate” (yes, they use that word – weird for something that is actually a matrix multiplication) them to the (pseudo-inverse) sector portfolio.

They do not seem to realize that they have forgotten the stock-specific risk in the process. It might be correct if your sector decomposition is coarser than their factor decomposition, but otherwise, it is plainly wrong.

Incidentally, they also forget to suggest what to do with missing values – stocks whose sector is unknown – I suggest to set their exposures to zero.

Finally, they forget to stress the absence of a formula to compute the risk from the contributions to risk – which should not be called “contributions”, after all:

$$\sum_{j \in \text{sectors}} (\mathbf{w}' f)_j \text{MCTR}(\mathbf{w}, \mathbf{s}_j) \neq \text{Risk}(\mathbf{w}).$$

Japanese Equity Model
Barra (1999)

Risk can be decomposed into:

- *specific risk*, associated with individual securities (unless you are a stock-picker, you try to avoid it);
- *common risk factor*, associated with industries and other factors (these correspond to the bets we make);
- *systematic risk*, associated with the market (you cannot avoid it – well, with cash, futures or with no long-only constraint, you can).

The *Capital Asset Pricing Model* (CAPM) is a no-intercept linear regression of the (excess) returns of a stock against the (excess) returns of the market; the

coefficient is called the *beta* of the stock or its *exposure* to the market.

Multi-Factor Models (MFM) are linear regressions of the (excess) returns of a stock against various “factors” (industry, market excess return, oil price, temperature, etc. – you have to choose your own factors).

Arbitrage Pricing Theory (APT) is the use of MFM to estimate the variance-covariance matrix of the stock returns – more generally, a *risk model* is a means of estimating a variance matrix.

There are several types of active investment strategies: *market timing* (sell before the market goes down, buy before it goes up); sector emphasis, stock selection.

Risk models can be used to create a portfolio, to analyze one (check that the risk is where the manager claims it is), or to attribute its performance to the various factors.

This document does not stress the difference between the Barra way of estimating a risk model and others: you always have

$$\text{returns of stock } i = \sum_{k \in \text{Factors}} \beta_{i,k} r_k,$$

but

- Barra estimates the exposures $\beta_{i,k}$ of stock i to factor k and computes the factor returns r_k ;
- Northfield estimates the factor returns and computes the exposures;
- APT estimates the factor returns from the first eigen vectors of the variance-covariance matrix and computes the exposures (this is called a *statistical risk model*: you then have to interpret the various factors – it is also linked to Principal Component Analysis (PCA) and Random Matrix Theory (RMT));
- USB uses a hybrid approach: they start with the factor returns, use them to estimate the exposures, use the exposures to estimate the factor returns, and so on, until convergence.

The document ends with a rather comprehensive glossary.

MNP: R package for fitting the multinomial probit model

K. Imai and D.A. van Dyk, Journal of Statistical Software (2005)

The multinomial probit model models a qualitative variable Y , with values 1, 2,..., n as follows:

W_1, W_2, \dots, W_n are (unobserved, latent) gaussian variables, of unknown mean and variance,
 $Y = \text{Argmax } W_i$

This package fits the model with an MCMC. Before interpreting the results, the article asks us to use the *coda* package to look at the Gelman–Rubin convergence statistic (numerically and graphically), to look at the time series of the sampled parameters, to look at the distribution of those parameters.

The appendix also explains how to set-up a “local library directory”.

BugXLA: Bayes for the Common Man **P. Woodward, Journal of Statistical Software (2005)**

This is an Excel interface to WinBugs – but they seem to completely forget the convergence problem...

Calling the lp_solve linear program software from R, SPlus and Excel **S.E. Buttrey, Journal of Statistical Software (2005)**

There is another, faster, more reliable R package for linear and mixed integer programming (MIP), *lpSolve*; it relies on the free (LGPL) *lp_solve* library.

The article insists on the Excel interface and the fact that it yields the right result, contrary to Excel.

...
 ??. ? et al. (????)

This article explains how to perform a “regression with time-varying parameters”. The main difference with my current implementation is:

- They assume the parameters are AR(1) processes, I assume they are random walks.

There are other, minor differences:

- They take sensible initial values for the estimation of the coefficients, I take 0;
- They give the p -value of a Likelihood Ratio (LR) test to check if the parameters are indeed varying (but to compute it, they have to know the “number of degrees of freedom” of the model with time-dependant parameters – I am not sure how they estimate it);
- Their Kalman filter seems to be univariate;
- They forget to check that their data look gaussian, that the residuals look gaussian and iid.

Random matrix approach to cross-correlations in financial data

V. Plerou et al., Phys.Rev.E 65 066126 (2002)

Another article on the same subject:

- The bulk of eigenvalues is consistent with RMT (they look at the distribution of eigenvalues, at the distance between two adjacent eigenvalues (nearest neighbour spacing), at the next nearest neighbour spacing – and another quantity I do not understand);
- There are no significant negative correlations;
- The largest eigenvalue represents the market;
- The next eigenvalue represents the logarithm of the capitalization;
- The next eigenvalues represent sectors (they provide a formula to estimate the number of significant stocks in a normalized eigenvector: the inverse of the sum of the fourth power of the coordinates);
- The smallest, non-significant eigenvectors correspond to correlated pairs;

- Eigenvectors with large eigenvalue are stable over time (for this, they compute the *overlap matrix*, whose (i, j) element is the scalar product of the i th eigenvector at time t with the j th eigenvector at time $t + T$; discarding the smallest eigenvalues, *i.e.*, using a (statistical) risk model to optimize a portfolio yields more reliable risk and returns forecasts.

Shortcomings in advise versus the art and science of investing in hedge funds
G. Susinno, 2005

Risk models are ways of reliably estimating variance-covariance matrices. *Statistical risk models* proceed as follows: compute the variance-covariance matrix, diagonalize it and zero out the lowest eigen values. *Random matrix theory* tells us how many eigen vectors to retain: compute the variance matrix obtained after shuffling each time series, repeat several times to get an idea of the distribution of the eigen values.

To visualize this correlation matrix, turn it into a distance matrix and draw the *Minimum Spanning Tree* (MST) – when distance analysis and clustering fail to bring any insight on the data, MST often prove enlightening and can highlight the structure of the data. (The author tries hard to impress the reader by using the words “ultrametric distance” – an *ultrametric distance* is just a distance measured on a tree.)

The author falls into (at least) two traps. First, even though he stresses the non-gaussianity of the data, the lack of *power* (he does not seem to know that word) of gaussianity tests, he uses the eigen value distributions for gaussian random matrices, instead of the permutation-based tests. Second, the permutation-based tests assume that in the initial time series, consecutive observations were independent – a suspicious assumption.

Risk premia on european sectors
K. Reimer and H. Berster, Commerzbank,
april 2005

One can also use the Residual Income Model (RIM) to compute the

risk premium = k – risk-free interest rate.

EROS (Expected Returns on Stocks)
K. Reimer and H. Berster, Commerzbank,
april 2005

They have 3 ideas:

- Use the *Residual Income Model* (RIM) to compute the “market expectation of returns” k , remark that it has a very high rank correlation with future returns (IC, Information Coefficient) and use it to optimize a benchmark;
- Compute a *value* measure, as a forcefully normalized expected *divident yield* over the next 5 years, corrected to account for the end of the fiscal year;

- Similarly compute a *growth* measure, as the annual *earnings growth*; Use those two measures to interpret the portfolio.

This “forceful normalization” has a drawback: it allows you to compare individual stocks or sectors to the whole market, but does not give you any information for the market as a whole – its growth and its value are, by definition, zero.

The succinctly recall how the Residual Income Model (RIM) works:

- If the risk-free interest rate is high, investors will only invest in companies with a higher prospective return: the price of the companies will depend on the current risk-free interest rate – more precisely, it will depend on the current “interested rate” or desired return, k , for investments providing a similar risk profile.
- The price of a given company will also depend on its Return on Equity (RoE).

The RIM sums this up:

$$\text{Fair Value} = BV_0 + \sum_{t=0}^{\infty} BV_{t-1} \frac{RoE_t - k}{(1 + k)^t}$$

where

BV_n = Book Value in n years
 RoE_n = Return on Equity in n years

The difference $RoE - k$ is called *abnormal earnings* or *residual income*.

Of course, one has to provide a mode for the evolution of the book value BV_n and the returns on equity RoE_n – the article does not give any detail and simply mentions a “three-stage model using IBES consensus forecasts”.

We know neither the fair value nor the desired return k , but if we replace the fair value by the current market price, we get a market estimation of the desired return k .

Longitudinal and Panel Data: Analysis and Applications for the Social Sciences,
E.W. Frees, Cambridge University Press, 2002

Chapter 1 presents and motivates the subject, longitudinal data: it is a mixture of regression and time series analysis:

- Instead of studying a single time series, we study several at the same time; *i.e.*, we have several realizations of a stochastic process.

- We investigate a relation $y = \alpha + \beta_1 x_1 + \beta_2 x_2 +$ noise that evolves with time: α , β_1 , β_2 may change with time. This could suggest a 2-step procedure: for each t , estimate the coefficients $\alpha(t)$, $\beta_1(t)$, $\beta_2(t)$; then study the resulting time series. Longitudinal data analysis is a similar but reliable, 1-step procedure. The noise is often called *subject-specific dynamic pattern*.

Chapter 2 lists the plots used to investigate longitudinal data:

- density estimation (histogram, etc.) of each variable and evolution with time (violin plot, lattice plot);
- $y \sim x_i$ for the whole period and its evolution over time (in a lattice plot or as paths on a single plot);
- Time series of each variable for each subject (or group of subjects);
- Added variable plots, discarding the effects of time $y_i(t) - \bar{y}_i \sim x_i(t) - \bar{x}_i$, of the other variables $\text{lm}(y+x_2+x_3)\$res \sim \text{lm}(x_1 \sim x_2+x_3)\res , or of both

the possible temporal covariance matrices (independent, AR(1), AR(T)) and lists a few tests: e.g., you no longer test the *influence* of an observation (as with the Cook distance, in classical regression) but that of a whole subject on the regression results.

Chapter 3 motivates *random effects* and *mixed effects models*: the observations are grouped (various subjects (stocks) are studied over time, thus we have several observations per subject) but the subjects do not represent the whole population: we want to take into account the variations between subjects and to be able to extend the results to the whole population – to this end, we provide a model for the *subject-dependant parameters*. Those parameters (aka *nuisance parameters* or *random effects*), are themselves random variables, e.g.,

$$\begin{aligned} y_{i,t} &= \alpha_i + \beta x_{i,t} + \varepsilon_{i,t} \\ \alpha_i &\sim N(\mu, \sigma) \\ \varepsilon &\sim N(0, 1) \end{aligned}$$

Random effects can be computed as a Generalized Least Squares (GLS: like Ordinary Least Squares (OLS), but the noises can be correlated and/or heteroskedastic) problem (but you have to know the variances of the parameters: estimate them as you can). Restricted Maximum Likelihood (REML) provides better, less biased results than ML – but for tests, only use ML.

Chapter 4 stresses the difference between *estimators* (of the fixed effects) and *predictors* (of the random effects); predictors are actually *bayesian* Maximum Likelihood estimators, because we have a *prior* distribution for them – or, at least, a model for that distribution.

Chapter 5 generalizes the previous models to (multi-level) hierarchical models; they are still estimated as a single GLS problem, with a more parsimonious covariance structure. One should beware of the tests of hierarchical models: the usual ones are no longer valid, especially tests about the covariance.

Predictive variables in a regression can either be *stochastic* (i.e., random variables) or deterministic (e.g., fixed by the experimenter). Chapter 6, very technical, details the properties those stochastic predictive variables should have for a mixed model estimation to be valid. Problems appear, for instance, when you include the lagged variables to predict as a predictor or when the predictors are autoregressive.

Chapter 7 lists a few dangers of panel data models:

- If the model is too complicated, different elements of the model can lead to similar effects, making the model *unidentifiable*;
- You should account for (or beware of) omitted variables (subject-dependant, sector-dependant, time-dependant);
- You should account for missing data (discarding missing observations or imputing them can lead to suboptimal results).

Chapter 8 tackles dynamic models: if you want to use your model to forecast future values, spotting changes in the model is pivotal. Dynamic changes can be included in the model in several ways:

- add the time as a predictive variable (this works fine if its effects are linear);
- use the discrete derivatives of the variables;
- Allow for serial correlations (AR(1), etc.) in the noise;
- Allow for time-varying parameters;
- Include the lagged variable to predict as a predictor;
- *adapt* the Kalman filter.

The authors details the Kalman filter example (with the CAPM), unfortunately hiding the ideas behind complicated formulas. The important thing to note is that you should *not* proceed in two steps, first computing cross-sectionnal regression, then applying a Kalman filter on the coefficients, but instead *adapt* the Kalman filter to the model.

Chapter 9 focuses on logistic regression (i.e., predicting binary variables): here, for technical reasons, we need to replace the Maximum Likelihood Estimation (MLE) by *conditional MLE*. This chapter also succinctly presents the Generalized Method of Moments (GMM), aka Generalized Estimating Equations (GEE) – if you do not want to estimate the random effects but just their first two moments.

Chapter 10 presents the General Linear Model (GLM) and delves into the details of the GMM/GEE method (forgetting to state the ideas and instead giving equation after equation). I still have not understood what it is. As an example, they apply the Conditionnal MLE to the Poisson regression.

Chapter 11 tackles the case of a qualitative variable to predict, which can be treated as a multinomial regression problem or as a Markov chain one. In the middle of this chapter, they also mention the generalized Extreme Value Distribution (EVD) – but I do not know why. The slides do not mention survival models, but they appear in the book.

Mixed-Effects Models in S and S-Plus
J.C. Pinheiro and D.M. Bates
Springer Verlag, 2000

This is the official documentation of the R `nlme` package, that deals with mixed data, such as:

- Grouped data: several observations of one (or several) variable(s) for each subject, the subjects being partitioned into groups (and the groups can themselves form groups of groups);
- Longitudinal data: one (or several) variable(s) observed several times for each subject, at different points in time;
- Panel data: idem, but the observation times are the same for each subject, so that you can put the data in tables, one subject per row, one date per column;
- Repeated measures: several observations for each subject;
- Blocked design: several experiments (“experiment factor”) on several subjects (“block factor”);
- Split-plot design: idem;
- Multilevel data: each subject is in a group, each group is in a group of groups, etc.

These models could be written as:

```
y ~ x | subject/group
y ~ time | subject
y ~ 1 | subject
y ~ experiment | subject
y ~ 1 | subject/group
```

Chapter 1 is an overview of mixed models. When facing a new data set, you often assume that the observations are independent. But this is not always the case: in particular when you have several observations for each “subject” (a stock, in finance, a patient, in medicine), the observations for a given subject will be correlated. The same problem occurs if the subjects form several groups: the observations in a given group will be correlated.

One solution is to consider that the parameters to be estimated depend on the subjects, e.g. (this is a *fixed effects model*),

$$y_{it} = \alpha_i + \beta x_{it} + \text{noise}_{it}$$

Estimate: $\beta, \alpha_1, \alpha_2, \dots, \text{Var}[\text{noise}]$.

Alternatively, we can consider that the α_i are not parameters but random variables, following iid gaussian distributions (this is a *mixed effects model*):

$$y_{it} = \alpha_i + \beta x_{it} + \text{noise}_{it}$$

Estimate: $\beta, E[\alpha], \text{Var}[\alpha], \text{Var}[\text{noise}]$.

This is often preferable because there are fewer parameters and we can meaningfully extrapolate to subjects that were not included in the study.

This is a partly bayesian point of view: we have a prior on α (but not on β , and we take a Maximum Likelihood Estimator, which is very unbayesian).

Chapter 2 presents the underlying theory. In the following expression,

$$y_i = [X_i | Z_i] \begin{bmatrix} \beta \\ b_i \end{bmatrix} + \varepsilon_i$$

$$b_i \sim N(0, \Psi)$$

$$\varepsilon_i \sim N(0, V)$$

y_i, X_i, Z_i are known and we want to estimate β, Ψ and V . The coefficient β is said to be a *fixed effect* while b_i is a *random effect* – the word “effect” means “estimated parameter”.

This can be done by Maximum Likelihood (ML),

Likelihood(variance structure, parameters | data),

which produces biased estimates, or with Restricted Maximum Likelihood (REML): the restricted likelihood is the likelihood with the parameters integrated out; we use it to estimate the variance structure, and then, we plug this back into the likelihood to estimate the parameters.

From a computational point of view, one can first use the *EM* (Expectation–Maximization) method, that considers the random effects as unobserved (missing) variables (it goes very fast to the neighbourhood of the optimum, but then converges very slowly) and then the good old *Newton–Raphson* method (it converges fast when it starts close to the optimum).

The parameters are asymptotically gaussian, so one can use AIC, BIC and Likelihood Ratio (LR) tests (but you had better only use them to compare two models if the fixed effects are the same).

Chapter 3 presents the R functions to explore grouped data, fit mixed models and assess those models. For instance,

`lme(y~x, random= ~ 1 | subject/group)`

fits a model in which the slope is a fixed effect, the intercept is a random effect that depends on the two variables `subject` and `group`. If we wanted both the

slope and the intercept to be random, we would have typed (remember that when you write a model, the intercept is always implicitly included: we need not mention it)

```
lme(y~x, random= ~ x | subject/group)
```

or

```
lme(y~x, random=list(subject=~x, group=~x))
```

If you want your data to be easily plotted, with functions from the `lattice` library, you can store them in a `groupedData` object, that contains the data (as a data frame), a display formula (e.g., `y~time|subject`), “outer factors” (i.e., groups of subjects, e.g., `outer=~sex` or `outer=~Variety*Year`) and “inner factors” (i.e., other qualitative predictive variables, e.g., `inner=~Treatment`).

You might also want to investigate the functions `gsummary`, `intervals`, `augPred`, `plot.lme`, `summary.lme`, `VarCorr`, `lmList`, `simulate.lme`, `balancedGrouped`.

Chapter 4 suggests a model building strategy:

1. Fit a usual regression without taking the `subject` variable into account; look at the residuals in each `subject` (with a boxplot): if they are significantly different, this might warrant a mixed effects model with the `subject` variable. but we still do not know which effects should be fixed and which should be random.
2. Fit a regression for each subject, with the `lmList` function; look which coefficients vary a lot between subjects, with the `plot(intervals(...))` function: they will be your random effects; you may also want to look at the pairplots of the estimated coefficients, to get an idea of the covariance structure Ψ of the random effects.
3. Fit the model and examine it: boxplot of the residuals for each subject (to see outlying subjects), residuals vs fitted value (to check if there is a difference between groups: if so, add this new grouping variable), quantile-quantile plots of the residuals; also remember to check for heteroskedasticity and correlation problems.
4. Do not only look at the residuals: similarly, look at the random effects: quantile-quantile plot, pairplot (to see if you should add a coarser grouping variable, as above; and also to check if your assumptions about the covariance matrix were not unreasonable).

Chapter 5 explains how the covariance of the random effects (the Ψ matrix) can be specified. There is first the correlation structure among the random effects (investigated with the pair plots mentioned earlier). If you do not specify anything, the computer assumes that the covariance matrix has no remarkable property. If they are independent:

```
random=list(subject=pdDiag(~x1+x2+x3))
```

Other examples:

```
pdIdent(~ x1+x2+x3)
pdDiag(~ Variety-1)
pdCompSymm(~ Variety-1)
pdBlocked(list(~1, ~x1+x2+x3-1))
pdBlocked(list(pdIdent(~1), pdIdent(~ Variety-1)))
```

You should also tackle heteroskedasticity (the V matrix); e.g., if the variance of the noise depends on the `group` variable, you can say

```
weights=varIdent(form= ~ 1 | group)
```

But the general situation is more complicated: to simplify it, we decompose the noise variance matrix V into a variance component (a diagonal matrix, that will account for heteroskedasticity in the noise term) and a correlation component (a positive definite matrix with 1s on the diagonal, that will account for the lack of independence of the noise term), and we specify them separately.

Here are a few examples, to be given to the `lme` function as a `weights` argument.

The variance increases linearly with age:

```
varFixed(form= ~age)
```

The variance is constant in each group:

```
verbvarIdent(form = 1|group)
```

The variance increases linearly with the fitted values:

```
varPower(fixed=.5)
```

The variance increases linearly with the fitted values for one group only:

```
varPower(form= ~fitted(.) | Sex,
          fixed=list(Male=.5, Female=0))
```

There are other functions: `varExp`, `varConstPower`, `varComb`.

And now, the correlation structures, to be fed to the `lme` function as a `correlation` argument.

Compound symmetry: two observations for the same subject have a 0.3 correlation.

```
corCompSymm(value=0.3, form=~1 |Subject)
```

A general correlation matrix:

```
corSymm(form = ~ 1 |Subject)
```

An AR(1) correlation structure (by default, the order is that in which the observations appear – see also the `ACF` function):

```
corAR1(0.8, form = ~ 1 | Subject )
```

A continuous AR(1) correlation structure (the coefficient will be positive):

```
corCAR1(0.8, form = ~ time | Subject )
```

An ARMA correlation structure:

```
corARMA( c(.8, .4), p=1, q=1,
          form= ~ 1 | Subject )
```

There are also spatial correlation structures (see also the `plot(variogram(...))` function:

```
corExp( 1, form = ~ x + y )
corGauss(...)
corLin(...)
corRatio(...)
corSphere(...)
```

This chapter also presents the `gls` function (Generalized Least Squares), to fit (non-mixed) heteroskedastic linear models – because *the correlation structure and*

the heteroskedasticity can have the same effects and therefore specifying both can lead to unidentifiability. People often wonder how to tackle heteroskedasticity in R: this is one solution.

Part 2 (I have not carefully read it) tackles non-linear models: you should use them when you know the mechanism producing the data (this is often the case in chemistry) or when the data has natural properties, such as an asymptote. Otherwise, stick to linear models – with splines or local regression, if needed.

One limitation of the `nlme` package is the fact that, if we have several grouping variables, they have to be nested – but you can imagine mixed models in which they are not.

Another limitation is that it does not include Generalized Linear Models (GLM) such as logistic or Poisson regression: for those, you might want to have a look at `glmML` in the `glmML` package for GLM with a random intercept; `GLMM` in the `lme4` package for mixed GLM via REML; `glmPQL` in the `MASS` package (idem); `gee` in the `gee` package.

The under-hyped `lme4` package contains other interesting functions, such as the `lmer` function (a revised version of the `lme` function), that allows for nested and crossed random factors. The syntax for the mixed effects is the “natural” one:

$y \sim (x1 \mid g1) + (x2 \mid g2)$

Incorporating estimation errors into portfolio selection: robust portfolio construction
S. Ceria, R.A. Stubbs, 2005

Modern Portfolio Theory (MPT) works as follows: estimate the returns and the volatility of each stock, then try to find the combination of stocks (“portfolio”) with the highest return among those whose volatility is within the limits imposed by your clients.

$$\begin{aligned} &\text{Maximize } \alpha^T w \\ &\text{Subject to } w^T Q w \leq v \end{aligned}$$

where α are the predicted returns, Q is the covariance matrix of the predicted returns, v is the maximum volatility allowed, w is the portfolio composition (to be found).

This is a *quadratic optimization problem*: the set of feasible solutions is convex, with sharp edges (sharp because of the linear constraints we do not mention) and the best solution will be on one of its vertices. When we slightly change the inputs, the solution may switch from one vertex to another.

Several solutions to overcome the imprecision of returns forecasts have been proposed:

- shrink them towards the average expected return (James–Stein);
- shrink them towards that of the minimum variance portfolio (Jorion);

- Adopt a bayesian approach (Black–Litterman);
- Increase the risk-aversion parameter (Horst);
- Resample (Michaud) – if you have time and no non-convex constraints, such as a limit on the portfolio size or a minimum investment per asset.

The idea of robust optimization is:

$$\begin{aligned} &\text{Maximize } \text{Min}\{\alpha^T w, \alpha \in B(\bar{\alpha}, r)\} \\ &\text{Subject to } w^T Q w \leq v \end{aligned}$$

where $\bar{\alpha}$ are the return forecasts,

$$B(\bar{\alpha}, r) = \{\alpha : (\alpha - \bar{\alpha})^T \Sigma^{-1} (\alpha - \bar{\alpha}) \leq r^2\}$$

is an ellipsoid, centered on $\bar{\alpha}$, containing the actual forward returns and α are the worst-case returns in this ellipsoid. The magic is that one can actually compute that minimum:

$$\text{Min } \alpha^T w = \bar{\alpha}^T w - r \|\Sigma^{-1} w\|.$$

We thus have a new convex optimization problem – but it is no longer quadratic: we need either a general convex optimization algorithm or a “symetric second order cone optimizer”.

This was the general idea, but it still has to be refined. Two problems are:

- We are too conservative, we have assumed that the errors were always in the worst direction;
- This approach only works for long-only portfolios.

We can introduce a “bayesian prior” (they claim they do not like bayesian methods, but it is what they are doing) as a new constraint, stating that the upward and downward errors should cancel each other (but you can refine this, to account for the larger number of overestimated returns and for the volatility).

Axioma, the future of risk analysis, rebalancing and trading analytics
D. Cashion (2005)

Mean-Variance Optimization (MVO) consists in taking a set of assets, with their predicted returns and estimated volatility and building the portfolios with a given volatility, the maximum expected return, satisfying a set of constraints (e.g., long-only, at most 2% in each stock, etc.). The problem is that the computer assumes that the predicted returns are accurate while they are not: if we change them a bit, the resulting portfolios can drastically change. The reason is that the possible weights of the various assets in the portfolio form a convex set, with sharp vertices, and the result portfolio is on one of those vertices: when we change the inputs, either we stay on the same vertex, or we switch to another one.

Instead, Axioma incorporate uncertainty in the inputs, they replace exact values by intervals (ellipses, actually) of possible values; this amounts to bevel the vertices of our convex set.

They also add discrete heuristic algorithms for rules such as “maximum number of holdings” or “minimum holding”. if-then-else rules, etc.

***The generalized dynamic factor model:
representation theory***
M. Forni and M. Lippi

Here are a few ideas to study panel data:

- Consider the time series one at a time – but this will not account for the correlation between the time series;
- Use a VAR model – but this will only work if you have few subjects: we have thousands of them;
- In the *dynamic factor analytic model*, aka *index model*, there are (latent) orthogonal variables $f_1(t), \dots, f_n(t)$ and, for each subject, the variable studied is a linear combination of those factors plus an *idiosyncratic component* (aka *subject-specific dynamic pattern*), orthogonal to the factors and pairwise orthogonal among them,

$$x_i(t) = \beta_1^i f_1(t) + \dots + \beta_n^i f_n(t) + \varepsilon_i(t)$$

- You can also add in the lagged factors;
- Finally, you can remove the requirement that the idiosyncratic factors be pairwise orthogonal to account for, e.g., shock propagation.

This last model is the *generalized dynamic factor model*.

But how do yo estimate the coefficients of the model? Well, you do not. However, a preceding article, by the same author, explained how to infer the number n of factors and this articles states that the common component $x_i(t) - \varepsilon_i(t)$ is the projection of the signal on the first n “*dynamic principal components*” – but I have not understood what it meant.

TODO: Explain “dynamic Principal Component”.

***Linear regression with errors in both variables:
a proper bayesian approach***
T.P. Minka, 1999

Linear regression focuses oin the relation between (say) two variables, e.g., $y = \alpha + \beta x + \text{noise}$: we assume that there is a simple, deterministic relation between x and y and that y is observed with errors. But what if x is also measured with errors? You can:

- Forget about those errors if you just want to predict y from the imprecisely measured x ;

- Try more symetric methods, such as Principal Component Analysis (PCA): it gives you some insight as tp the relation between x and y , put focuses on the joint distribution $p(x, y)$, not the conditional distribution $p(y|x)$ – do not use it for predictions;
- Account for those errors with an EIS (Error In Variables) regression: we have a relation $y_0 = \alpha + \beta x_0$ but we can only measure $y = y_0 + \text{noise}$ and $x = x_0 + \text{noise}$ and we want to find α and β – we do not want to make forecasts, just to study the mechanisms behind the data.

The article presents three EIS estimators:

- TLS (Total Least Squares): write the likelihood (which depends on α, β and x_0), replace x_0 with its most probable value and find α and β by the values that maximize the likelihood;
- Maximum Likelihood (ML): integrate x_0 out of the likelihood and maximize this marginal likelihood;
- Posterior Mean: the posterior distribution tends to be bimodal: the ML estimator is not stable; we replace it by the expectation of the posterior distribution of (α, β) .

***Liquidity and autocorrelations in individual
stock returns, D. Avramov et al., 2005***

When an investor wants to buy or sell a large amount of a given stock, the price will rise or drop for a short moment and then revert to normal. Seemingly profitable contrarian trading strategies can stem from this phenomenon. To check wether they are profitable, the article combines

- Past returns;
- Illiquidity (defined as

$$\frac{|\text{returns}|}{\text{traded value}}$$

or as the average ask/bid spread);

- Turnover

to define their strategy, equally weighted or “relative strength”-weighted (more weight on the stocks whose recent returns were high) and estimates the transaction costs (in three ways, but the only detailed one is the ask/bid spread). In the end, the strategy is not profitable, the Efficient Market Hypothesis is not violated.

In other words: “non-informed trading is accompanied with high trading volume, informed trading is accompanied with low trading volume” – “changes accompanied with high volume revert”, “changes accompanied with low volume need not revert”.

**Panel data models for stock returns:
the importance of industries
R. Bauer et al, LIFE, 2003**

In finance, to study a factor, *i.e.*, a variable suspected of providing some insight into future returns, one usually uses the non-parametric (“non-parametric means that it does not rely on statistical assumptions that one could forget to check – it is fool-proof”) *portfolio method*: build five portfolios, each containing the stocks in one quintile of the variable and monitor their returns. This can be generalized to two factors – but not more: beyond that, many fractiles would be empty...

Instead, in order to include more variables, people have tried regression (often forgetting that there are assumptions to be checked, at least graphically, before performing the analysis, and forgetting to assess the quality of the results, *e.g.*, with diagnostic plots). But financial data is inherently bidimensional, *i.e.*, the variables we play with do not form vectors but rather 2-dimensional arrays, one stock per row, one date per column. One can forget this 2-dimensional structure and compute the usual (“Ordinaly Least Squares” or OLS) regression. But by doing so we discard some of the information available.

Instead, we can slice the data and consider one date (column) at a time, compute the regression coefficients for each date (people say “cross-sectionnally”) and take the mean of those regression coefficients: this is the *Fama-McBeth regression*; it takes into account the unbalancedness of the data.

But this suggests that the quality (variance) of the estimation of the regression coefficients is the same for each date – this need not be the case. We can estimate this variance and take it into account by computing a weighted average of the coefficients (the weight of a given coefficient at a given date is the inverse of its estimated standard deviation): this is called *weighted regression* or *weighted least squares*.

But we still have a problem: we assume that these regression coefficients are constant over time – this is surely wrong. To take this effect into account, we must leave the realm of DIY statistics and enter that of *mixed models*. We can then consider regressions of the form

$$\text{return} = \alpha + \beta \cdot \text{factor} + \text{noise}$$

where the regression coefficients α and β may themselves depend on other data, say:

$$\alpha \sim \text{stock} + \text{industry} \times \text{time}$$

$$\beta \sim \text{industry}$$

$$\text{noise variance} \sim \text{time}.$$

This is the example detailed in the article (with two differences: they have a dozen factors instead of one and they assume that the variance is constant), but you may change it as you want.

The numerical estimation of the parameters of those mixed models used to be very intricate (that is why

some people still use “Fama-McBeth regression” or “weighted least squares”) but, luckily, R can now do that for us.

The article also remarks that statistical tests with a 5% confidence level are useless in this context, because the huge volume of data leads to very small p -values. Instead, they use the Schwartz Information Criterion (SIC). We already mentioned this problem when we studied Bayesian Model Averaging (BMA) – we had used the Bayesian Information Criterion (BIC), aka Schwartz Bayesian Criterion (SBC).

**VWAP reversion
S. Usui, Nomura, 2004**

They propose the following (intraday) investment strategy:

- If $\text{index} \gg \text{15-minute VWAP} \gg \text{60-minute VWAP}$, then long the index;
- If $\text{index} \ll \text{15-minute VWAP} \ll \text{60-minute VWAP}$, then short the index.

where VWAP is the volume-weighted average price – it is a replacement of the moving average, better-suited for intra-day data. This is just a refinement of the strategy:

- If $\text{price} \gg \text{Moving Average}$, then buy;
- If $\text{price} \ll \text{Moving Average}$, then sell.

that relies on the belief that the price is mean-reverting – the refinement selects periods when the price has a significant upward (respectively downward) trend, visible both on the 15-minute VWAP and the 60-minute VWAP.

They motivate this strategy by a linear relation between (price – 15-minute VWAP) and (15-minute VWAP – 60-minute VWAP). Here, of course, regression is a bad idea: first the data are not normal (they are fat-tailed), second, regression is asymmetric (you try to predict one variable from the other) while the strategy is symmetric (if you want a plot, draw the first principal component, if you want a test, a simple correlation test would do).

They try to tune their strategy by modifying the thresholds above which those inequalities trigger the trades, but:

- They do not tune the time parameters (15 minutes and 60 minutes);
- Their tuning is not adaptive.

Trading Securities Using Trailing Stops
P.W. Glynn and D.L. Iglehart,
Management Science, 41:6 (1995)

To avoid losses, one can sell whenever the price (of a stock you have bought) goes to far away (say, \$5) below the previous maximum. The article investigates, from a theoretic point of view, the time you hold the asset, the return, the “best” limit to choose – under the unrealistic assumption that the price follows a random walk with a constant linear trend, oblivious of the fact that those stop-loss rules are designed to get rid of stocks that do not follow our model or to spot a change in the model.

High frequency data filtering
T.N. Falkenberry, TickData Inc. (2002)

High-frequency data may be unclean, even more so than traditional data because the sheer volume of data prevents their efficient automatic (or manual) detection.

The “errors” can be: isolated bad ticks, bad ticks in succession or real outliers (to be retained, otherwise we would build a model on data with an unrealistically low volatility).

The errors may be obvious at some scale but not at larger scale.

To automatically detect some of those errors, you may remark that:

- The tick frequency depends on the capitalization and on the volume;
- The tick frequency depends on the time of the day and the capitalization (constant accross the day for small-caps, constant accross the day except for an opening peak for mid-caps, U-shaped for big-caps).

The filter they provide takes into account: those different tick frequency profiles, changing volatility, overnight gaps.

They do not give any detail, but the filter could look like:

- Compute a MA and a moving standard deviation; start anew every morning to accommodate overnight gaps; if the standard deviation is too low, set it to 2 ticks;
- If a point is more than 3 standard deviations from the moving average, tag it as a bad tick; you may want to change the threshold;
- If the last tick was a bad tick and you would like the current tick to be a bad tick as well, don’t, and reinitialize the moving average and the moving volatility; you may want to modify this rule to accommodate repeated bad ticks (they would be equal);

- Replace bad ticks with ticks with the same volume and the preceding price.

Predicting Returns with Financial Ratios
J. Lewellen, 2003

To test if a factor x has a predictive power on the returns r , one usually uses Ordinary Least Squares (OLS, aka “classical regression”), even if the method’s hypotheses are not satisfied. But the model is often:

$$r_n = \alpha + \beta x_n + \varepsilon_n$$

$$x_{n+1} = \phi + \rho x_n + \eta_n$$

$$\text{Cor}(\varepsilon, \eta) = \gamma.$$

With just the first two equations and independent noises, OLS regression and tests give correct answers, but with the last equation, the results are biased towards predictability.

The article, oblivious of other methods, such as Maximum Likelihood Estimators (MLE), tries to *force* OLS into that framework and derive reliable tests.

Natural SelectionTM: Matching strategies to assets
Evolutionary Finance Ltd, June 2003

They present an investment strategy design software, based on “patented” genetic algorithms (it seems to be a UK company: software patents are illegal in Europe...), that proceeds as follows.

- Take a set of possible investment strategies, each yielding a “buy/sell” signal (they give an extensive list of such signals);
- Combine those strategies to find the “best mixed strategy” at a given time, for a given universe, for a given horizon;
- Given the history of those “best” strategies, try to predict the next best strategy.

There are a few problems:

- They do not explain how to *combine* strategies – this is the heart of the algorithm.
- Using *the* previous best investment strategy to predict the next best one means that you discard all the information about strategies almost as good as the best – the actual best strategy is sample-dependant, you surely do not want to discard all the information contained in the near-best strategies.
- They do not explain how they use genetic algorithms in the second step, to predict the next best strategy; one could use straightforward bayesian methods or a Markov chain: is there more than that?

Backtesting a strategy often suffers from a serious bias: it selects a strategy adapted to the backtest period, not to the present or to the future. The author advocates methods able to detect *regime shifts* – such as their evolutionary models.

***Detecting a currency's dominance or
dependance using foreign exchange network
trees***

M. McDonald et al. arXiv:cond-mat/0412411

This article applies “*distance analysis*” fo financial data. There are a few differences fromn the classical, idealistic situation you might be used to.

- Oftentimes, distance analysis simply does not work. In order to actually see something, you can compute and plot the *Minimum Spanning Tree* (MST) of the data. This is actually a crude, 1-dimensional form of “local distance analysis” – if you want more about this, have a look at the *isomap* algorithm.
- There are more missing data than usual. The distances are computed from a correlation matrix (there are several ways of doing so, e.g., $d = \sqrt{1 - \text{cor}}$ or $d = \sqrt{1 - |\text{cor}|}$). When you compute a correlation matrix, you have to remove all the observations (here, the dates) for which at least one value is missing – it you just remove the pairs for which one value is missing, you might not get a positive definite matrix.
- As the data are exchange rates, both USD/EUR and EUR/USD are present: with the chosen distance, the two points will be very far apart.
- The fact that the product of the exchange rates of any three currencies, say USD/EUR, EUR/JPY, JPY/USD, is 1 imposes some structure on the tree. To see if this structure dominates or if the resulting tree contains more information, they suggest to resample one (only one) of the currencies and to compare the two trees.
- They also check, visually, that the relations between the various exchange rate returns are linear.
- They check if lagged correlations contain some information: for hourly exchange rate data, there is none. If there were, we would depict the corresponding relations by a *directed* graph.
- They also investigate the sability of the tree over time (by looking at the edges that remain).

I did not read the book but just three extra “chapters” available on the Web (<http://www.bio.ic.ac.uk/research/mjcraw/statcomp/chapters.html>). The first one is about *gamma regression*, i.e., linear regression with (non normal) gamma-distributed noise: you use it when the relation looks linear but the noise is asymeric – however, the motivation and example they give could be dealt with a simple transformation ($1/y \sim 1/x$) – quite confusing.

The second chapter, about Generalized Additive Models (GAM) was superficial but fine.

The third chapter was the most confusing: they state that PCA and Factor Analysis are different but give the same interpretation (why, then, are the results different? Which method should we choose in which situation?); they mix up *supervised learning* (in *discriminant analysis* we know the classes beforehand we do not try to find if there are classes, how many there are or how to interpret them, as the author sometimes (but not always) seems to think) and *unsipervised learning*.

Neither buy nor even read that book!

Global Financial Data Products
M. Bulsing and A. Scowcroft
UBS Investment Research, April 2002

A badly written article (the abstract is too long, contains too many technical terms; there is no distinction between the data that come from the data sources and the data that has been computed from the data sources; etc. – the preceding article used marginal notes to improve readability: this was a good idea) that tries to explain what can go wrong when you handle financial data:

- multiple data sources, with multiple conventions;
- data cascades (you take data from one source, perform some computations, combine the result with another data source, perform some more computations, etc.);
- mis-aligned data: “monthly data” can be “data at the end of the month”, “data at mid-month”, “data smoothed across the month” – furthermore, corporate events might occur within the month;
- Market capitalization requires the “number of shares”, which fluctuates daily (because of options, for instance): data sources update this more or less continuously;
- Currency problems (not read);
- Models: as data providers, they try to add some value to the data they provide, by performing some computations, based on models; they must

include enough variables in those models but not too many.

They favour a model with:

- the market return
- a factor (“yield”), for which they provide several formulas, each corresponding to a given “style” – they state the problems posed by each factor;
- the sector;
- the region (considering the country would lead to too many classes).

They do not mention the problem of missing data, of outliers, of errors and sanity checks, of unsatisfied normality or linearity assumptions in their regressions.

Active Return is Ambiguous

M. Partridge at al.

UBS Investment Research, August 2003

There are several definitions of “active return” (*i.e.*, “difference” between the return of your portfolio and that of a benchmark), but the most widely used have a few problems:

- They are not transitive: if they say that portfolio A is better than portfolio B and that portfolio B is better than portfolio C, they need not say that portfolio A is better than portfolio C;
- They can be incoherent and state that portfolio A is better than portfolio B *and* that portfolio B is better than portfolio A;
- They do not give the same result if you use monthly or daily data: they can say that portfolio A is better than portfolio B if you look at monthly historical prices and the opposite if you look at daily prices (these are fixed portfolios, not investment strategies: the portfolios are not rebalanced).

The definitions are as follows (active return, index dif-

ference, compounded return, log-return, index ratio):

$$\begin{aligned}\alpha_{\text{active}}(A - B) &= \sum_t \left(\frac{A_{t+1}}{A_t} - \frac{B_{t+1}}{B_t} \right) \\ \alpha_{\text{diff.}}(A - B) &= \prod_t \frac{A_{t+1}}{A_t} - \prod_t \frac{B_{t+1}}{B_t} \\ &= \frac{A_{\text{end}}}{A_{\text{begin}}} - \frac{B_{\text{end}}}{B_{\text{begin}}} \\ \alpha_{\text{comp.}}(A - B) &= \prod_t \left(1 + \frac{A_{t+1}}{A_t} - \prod_t \frac{B_{t+1}}{B_t} \right) \\ \alpha_{\text{log}}(A - B) &= \sum_t \log \frac{A_{t+1}}{A_t} - \sum_t \log \frac{B_{t+1}}{B_t} \\ &= \log \left(\frac{A_{\text{end}}}{A_{\text{begin}}} / \frac{B_{\text{end}}}{B_{\text{begin}}} \right) \\ \alpha_{\text{ratio}}(A - B) &= \left(\prod_t \frac{A_{t+1}}{A_t} \right) / \left(\prod_t \frac{B_{t+1}}{B_t} \right) - 1 \\ &= \frac{A_{\text{end}}}{A_{\text{begin}}} / \frac{B_{\text{end}}}{B_{\text{begin}}}\end{aligned}$$

The log returns or the index ratio have no problems (except, of course, being non-additive accross portfolios), and should be used instead of the others. The article pairs the pathologies and the indices (table 10 page 19).

The conclusion could be: if you want to add returns month after month, choose the log-returns (but do not add the log-returns of individual stocks to hope to get that of a portfolio), if you want to “add” the returns of stocks to get that of a portfolio, use the active returns (but *never* add them month after month). If you hesitate, choose the log-returns.

Wavelet compression, determinism and time series forecasting

I. Kaplan, 2003

The author presents a new way of assessing the noisiness (or volatility) of a time series: the percentage of compression (say, on a 64-element moving window) obtained by *lossless* wavelet compression.

Let us first recall what wavelet compression is. We start with a time series and we want to approximate it with a “simpler” function (yes, it sounds like regression). We first try to approximate it by a constant (a horizontal line); then, we take the residuals, cut them into two parts (the first half of the series and the last half) and try to approximate each part by a constant. And we iterate.

At each step, instead of approximating the data with a horizontal line, we can use a more complicated function (a line, a single sine wave – there are other less classical functions, called wavelets, with better theoretical properties).

This can be interpreted as: at each scale, a good forecast of x_n is $\frac{1}{2}(x_{n-1} + x_{n+1})$.

The idea of *lossless* compression is to start with an integer-valued time series, allow only integer coefficients and retain all the coefficients.

One can measure the “complexity” or the “compressibility” by counting the number of bits in the resulting set of *integer* coefficients (the binary code of an integer n requires $\lceil \log_2 n \rceil$ bits). Low values means low noise and high predictability.

Remark: lossy compression removes the low-amplitude noise and thus cannot estimate the overall amount of noise.

Remark: the author notes that the compressibility is reliably approximated by the number of bits of the first difference, which is easier to compute ($\text{sum}(\log(\text{diff}(x))/\log(2))$, in R).

Signal processing for everyone G. Strang (2001)

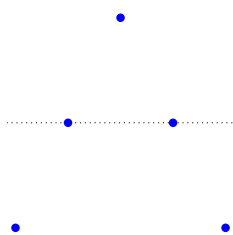
There are three kinds of Fourier transforms:

- The Fourier series expansion, that turns a (periodic) function into a sequence of coefficients;
- The Fourier transform, that turns a function (in L^2) into another function;
- The Discrete Fourier Transform (DFT), that turns a discrete signal (a sampled function, a sequence) into a periodic function (a function on the circle \mathbf{S}^1),

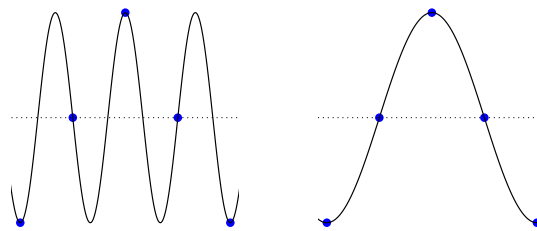
$$(x_n)_{n \in \mathbf{Z}} \mapsto X(\omega) = \sum_n x_n e^{-in\omega}.$$

The sampled signal and its DFT contain the same information, they are just two different ways of looking at the same data; when we look at the sampled signal, we say that we are in the *time domain*, when we look at its DFT, we are in the *frequency domain*.

In the real world, you never study a continuous signal: you sample it at discrete points and study the resulting sequence – hence our interest in the DFT. But we might lose information if the sampling frequency is too low: for instance, the following sampled signal



could come from one of the following continuous signals



this is called *aliasing*. We usually assume that the signal has no components with a frequency higher than half the sampling rate (or, equivalently, that we sample at a frequency higher than twice the highest frequency in the signal – this is called the *Nyquist frequency*: anything above that is lost by the sampling).

It would be nice if we could recover the signal from its samples, if we could interpolate it – under the assumption that the signal has no component of frequency higher than the sampling frequency, this is reasonable. One would expect a formula of the form $x(t) = \sum a_n g_n(t)$ where the a_n are coefficients computed in a complicated way from the sampled function and the g_n are functions – actually, things are even simpler: the coefficients in *Shannon’s formula* are the sampled values themselves,

$$x(t) = \sum_n x(nT) \cdot \text{sinc} \frac{t - nT}{T/\pi}$$

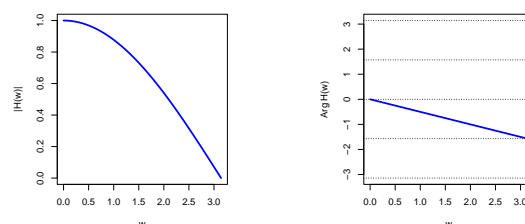
where sinc is the *sine cardinal* function,

$$\text{sinc } t = \frac{\sin t}{t}.$$

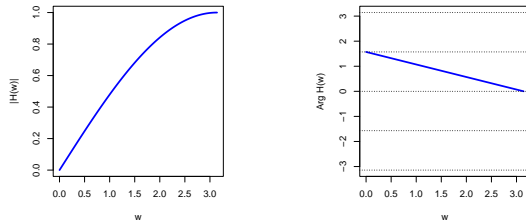
A *filter* is a way of transforming a sampled signal into another sampled signal. To study a filter, one first applies it to simple signals such as: a constant, a linear function, an impulse, an alternating sequence or a pure sinusoidal signal $x_n(\omega) = e^{in\omega}$. The filters may (or not) have the following properties: linear, time-invariant, of finite length, low-pass (the low frequencies are (mostly) preserved, the high-frequencies are (mostly) discarded), high-pass, causal (they do not peer into the future), they can preserve constant signals, linear signals, etc.

Linear Time-Invariant (LTI) filters are linear filters that do not change the frequency of pure sine waves, *i.e.*, $x_\omega \mapsto H(\omega)x_\omega$, *i.e.*, a sine wave is only changed in its amplitude and phase, *i.e.*, the filter is diagonalizable in the basis of pure sine waves. The function $H : \mathbf{S}^1 \rightarrow \mathbf{C}$ that describes the changes in amplitude and phase is called the *frequency response*.

We can now examine a few filters. First, the Moving Average (MA) is a lowpass filter,

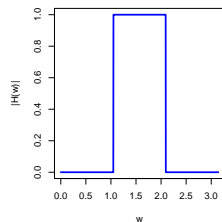


the Moving Difference is a high-pass filter,



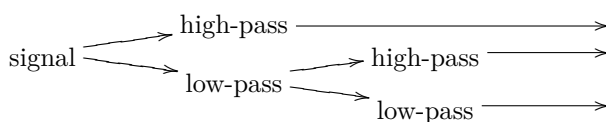
A linear filter can be seen as a convolution; in the frequency domain (recall that it means that we replace the sampled signal by its Discrete Fourier Transform), this becomes a multiplication – multiplication by the frequency response $H(\omega)$.

The article then becomes more complex and explains how to *build* filters. Ideally, we would like to have filters that look like



but this is impossible with finite-length filters: the best filter we can design will show ripples around the transition band, either very high but localized ripples (Gibbs phenomenon) or (better), ripples of the same moderate amplitude everywhere (equiripple filter, Remez algorithm).

The last part (two thirds) of the article focuses on *wavelet transforms*, that can be implemented as a *filter bank*.



The signal is divided into high and low frequencies, the low frequencies are divided into... (we do not divide the high frequencies further: they contain mainly noise – uninteresting information – but we could: we would get *wavelet packets*).

Bayesian model selection in social research **A.E. Raftery (1995)**

(This is the article accompanying the `bicreg` code – a very good and readable introduction to bayesian methods.)

Bayesian testing has the following advantages:

- It works with large samples. In classical hypothesis testing, the choice of the significance, α , is based on empirical remarks for samples of 30 to 100 observations that suggest $\alpha = 0.01$ to 0.05 ;

but what if our sample is larger? We should decrease α (we never do), but how do we balance significance and power?

- It is simpler: instead of having to consider two numbers, significance (α) and power (β , which is often forgotten), we only have one (the Bayes factor).
- It allows multiple tests, e.g., for model selection; with p -values, we have to transform the p -values, using the Bonferroni correction (designed by someone who assumed the observations were independent and who did not understand the notion of percentage but well-suited for people with a 4-operation (no “power” operation) calculator) or the Sidak one (that assumes that the observations could be independent): this is pretty awkward.
- It handles model uncertainty, we do not have to choose a single best model if none stands out.
- It can compare non-nested models.
- It is symmetric: the answer of classical hypothesis testing is either “we reject H_0 ” or “we do not reject H_0 yet”; the answer of bayesian hypothesis testing is “there is weak/strong/very strong evidence for H ”.

The main problem with hypothesis testing is the choice of the prior distributions. However it only contributes as $O(1/n)$ of the final result: for large samples, it is not important.

We can compare two models with:

$$\frac{P(\text{Model}_1|\text{data})}{P(\text{Model}_2|\text{data})} = \frac{P(\text{data}|\text{Model}_1)}{P(\text{data}|\text{Model}_2)} \times \frac{P(\text{Model}_1)}{P(\text{Model}_2)}$$

i.e.,

$$\text{posterior odds} = \text{Bayes factor } B_{12} \times \text{prior odds}.$$

They also detail (with heuristic non-mathematical arguments) the BIC approximation:

$$\log P(\text{data}|\text{model}) = \text{maximum likelihood} - \frac{d}{2} \log n + O(1)$$

where d is the number of parameters in the model and n the number of observations. This yields

$$2 \log B_{12} \equiv \chi_{\text{LRT},12}^2 - \text{df}_{12} \log n$$

where $\chi_{\text{LRT},12}^2$ is the likelihood ratio test statistic (aka *deviance*, it is the difference of the loglikelihoods of the models) and $\text{df}_{12} = d_1 - d_2$.

Actually, we have two definitions of BIC, that only differ by an additive constant:

$$\text{BIC} = \chi_{\text{LRT}}^2 \text{ with the saturated model} - d \log n$$

$$\text{BIC}' = \chi_{\text{LRT}}^2 \text{ with the null (empty) model} - d \log n$$

Here are a few rules of thumb to interpret the Bayes factor or the BIC difference:

BIC difference	Bayes factor	Evidence
2–6	3–20	positive
6–10	20–150	strong
> 10	> 150	very strong

One can convert BIC differences into p -value thresholds, but this depends on the sample size (e.g., strong evidence: $n = 20$, $\alpha = 0.05$; $n = 100$, $\alpha = 0.01$; $n = 10000$, $\alpha = 0.001$ – they give a complete table).

The article also details Occam’s window for model selection (strong version: retain the data-supported and parsimonious models; symmetric version: retain the data-supported models – the symmetric variant performs better).

The bayesian (*i.e.*, BIC-based) approach may also be useful when we want to select *one* best model: it works better (out-of-sample) than p -values or stepwise regression.

The conclusion mentions the model uncertainty problem and BMA – which surpasses bootstrap (bagging).

I have not read the comments after the article.

While reading articles about bayesian methods, you may stumble upon the following vocabulary.

- Non-informative prior: When choosing the prior distribution, we sometimes want to include as little information as possible in it (because we do not have this information). We may choose, for instance, a Cauchy distribution with a large dispersion parameter.
- Improper prior: Sometimes, we would even like to choose a “uniform distribution on \mathbf{R} ”: this is not a probability distribution, but it may work. A non- L^1 “density” function is called an improper prior. (Actually, it does not work that well: prefer a proper prior.)
- Conjugate prior: The conjugate family of priors is a family of probability distribution functions for which the computations are tractable – this depends on the problem at hand, more precisely on the likelihood function – if you do not do the computations by hand, forget about this notion.

Stock return predictability: a bayesian model selection perspective
K.J.M. Cremers

The author tackle the variable selection problem (for a 15-variable regression model of stock returns) with a bayesian approach.

Contrary to other articles that dismiss the issue of choosing the prior distributions by taking a “diffuse” one, this article examines

- the interpretation of those prior distributions, as

the investor’s belief (or not) in the predictability of the returns;

- the effects of these prior distributions on the results: there are no major qualitative differences, in both the skeptic and confident case, the posterior distribution suggests the returns are predictable from half a dozen variables.

They also performed out-of-sample tests: choosing the best model brings no reliable forecast, while a Bayesian Model Average (BMA) has a very small predictive power.

This could be improved by removing some restrictions (normality, linearity, parameter stability) and introducing lagged variables.

Choosing factors in a multifactor asset pricing model: a bayesian approach
J. Ericsson and S. Karlsson

Bayesian modeling goes as follows: you have a model whose parameters you want to “estimate”; you start with an a priori distribution of the parameters; you use the data to compute the a posteriori distribution of the parameters given the data. You can have several different models if you choose an a priori probability for each of them. The differences with the *frequentist* approach are:

- You have (and use) an a priori distribution on the parameters and/or models (if you do not have this information you may cheat and use a *diffuse prior*);
- The result is not an estimation of the parameters (that would be a single number) but the distribution of these parameters – the classical estimates may be obtained from this distribution, by taking the expectation (unbiased estimator) or the mode (Maximum Likelihood Estimator) – but the whole distribution provides more information.

This article is less badly written than the preceeding. They consider the variable selection problem in the linear regression of a stock return (or the returns of N stocks) with respect to several economic variables and several benchmark returns. The bayesian approach (with diffuse priors) gives a probability for each of these models. The interesting point is not (as the authors claim) the ability to choose the best model, but the ability to detect *model uncertainty*: indeed, the probability of the best model can be pretty low. I can see several causes to that model uncertainty:

- Some of the models are *very* similar or, at least, give similar predictions, and we do not have enough data to distinguish between them (this is the theoretical foundation of Bayesian Model Averaging);

- The best models are different but wrong, by the same amount (this is reality).

The article also details the 15 factors used in the study.

Thick Modeling
C.W.J. Granger and Y. Jeon

Extremely confusing article whose authors may not be as competent as they ought to be:

- In their “portfolio example”, they do not say where the data is, where the model is, which parameters are estimated. Actually, they consider several stocks, each known by the time series of its returns (this is the data). They assume that the returns are normal and independent and that their covariance structure is simple and can be described by three parameters (this is the model). They also define the quantity of interest: the composition of the optimal portfolio one can build with these stocks.

But their confusion goes further: this is not Bayesian Model Averaging (we would need *several* models) but a classical bayesian method: we have an a priori distribution of the model parameters, we use the data to compute their a posteriori distribution and then the a posteriori distribution of the quantity of interest.

- They confuse “noise” (what appears in a model) and “residuals” (what appears at the end of a regression – the residuals are an *estimation* of the noise).
- They seem to think that Maximum Likelihood can be used to compare models: you only use it when facing a *single* model (or for tests, with *nested* models).
- They suggest to perform Bayesian Model averaging with an equally-weighted trimmed-mean, as if they had never heard of MCMC.

***FlexMix: a general framework
for finite mixture models
and latent class regression in R***
F. Leisch (JSS 2004)

The EM (Expectation Maximization) algorithm is used to fit models with missing data, such as finite mixture models (for instance $z \sim x \cdot y$, where only x and y are known and z is qualitative). It works as follows:

- Choose an a priori estimate of the model parameters;
- (E-step) Replace the missing values by their expected values, assuming that the model parameters are correct;

- (M-step) Estimate the parameters by MLE (Maximum Likelihood Estimation), assuming that the missing values have the value inferred at the previous step;
- (E-step) ...
- (M-step) ...
- etc.

There are already some R packages to do that:

- `mclust` (for mixtures of gaussians);
- `fpc` (for mixtures of regressions);
- `mmlcr` (for mixtures of latent-class regression).

FlexMix encompasses the general linear models (Poisson regression, etc.) and is more configurable: you can provide your own model and write your own M-step.

iPlots
S. Urbanek and M. Theus (DSC 2003)

Interactive graphics should provide *colour brushing* and *linked highlighting*: the authors provide a framework to do so, for simple graphics (histograms, barcharts, scatterplots), implemented with SJava (but the user need not know anything about Java). It might be easier to use than GGobi (and prettier), but it lacks its multi-dimensional capabilities.

Synergy, issue 3, 2000

The Coil Contest is the European equivalent of the KDD Contest (but the KDD Contest is still alive). The conclusions of that issue are that for real-world problems, they suggest to try different algorithms (“look beyond your pet algorithm”) and to use a rigorous methodology (you can afford a crappy algorithm, but not a crappy methodology).

This issue also contains an article about wavelets, focusing on the combination of wavelets and “soft” algorithms (fuzzy logic, neural networks, genetic algorithms).

Do not expect to learn anything reading this – they provide interesting real-life examples, though – even one with high-frequency financial data.

***Using Perl for Statistics: Data Processing and
Statistical Computing***
G. Baiocchi (JSS, 2004)

This long and detailed article presents Perl as a glue language for statistics, navigating between remote data (web), statistical libraries (R, various Perl modules) and presentation tools (L^AT_EX), performing tasks such as data retrieval, transformation, validation and recoding. They suggest using Windows (why?) and Emacs (or SciTE, which also exists under Windows).

They list and assess the quality of various Perl statistical modules: reproductibility, reliability, benchmarking – the main problem is the Random Number Generator (RNG): avoid the default one and prefer MT or TT800.

They also mention matrix handling modules, PDL (a Perl package allowing one to manipulate matrices as easily as if they were scalars – like Matlab or R –, whose large graphical capabilities are not mentionned) and even Win32::OLE (to talk to the R DCOM server).

As many people claim to be a Perl guru after just a few hours, as the articles covers a lot of subjects, one could have feared that the author does not master some of them – this is not the case. It seems to be a good presentation of Perl, aimed at statisticians.

Kernlab – an S4 package for kernel methods in R

A. Karatzoglou et al. (JSS, 2004)

There are three R packages dealing with kernel methods: **e1071** (which uses libsvm), **klaR** (which uses SVMlight) and **kernlab**, with which you can add new kernels in R (with no need to dive into C or C++ code) and new algorithms (though many are already implemented).

The article starts by recalling the notion of S4 classes (with S3 objects, the class is merely an attribute, only used for method dispatching; with S4 classes, all objects of a given class have the same “slots” and methods are attached to the class – but if you just want to use S4 classes, the main difference is that \$ became @) and namespaces. It then lists the kernels and algorithms:

- SVM regression, classification (with 2 or more classes, but also with just one class: “novelty detection” or “outlier detection”);
- *Relevance Vector Machine* (a bayesian, weighted SVM regression);
- “*Gaussian processes*” (?);
- A Google-type ranking algorithm based on a network constructed with the kernel distance;
- kernel PCA, kernel CCA, clustering based on a kernel PCA.

The package also provides an *interior point quadratic optimizer* (most optimization algorithms (e.g., the *simplex* algorithm) examine the boundary of a convex subset of a high-dimensional space, because we know that the solution is on this boundary; but in high dimension, the boundary is huge and some algorithms are trying to take a “shortcut” *through* the convex: they are called “interior point” algorithms).

Wavelets in Statistics: a review

A. Antoniadis (1997)

This document is rather old and assumes that the reader is familiar with the theory behind wavelets (Besov spaces, etc.). Yet, the conclusion was enlightening, presenting “recent or future” developments (with no details):

- Non-parametric tests;
- Regression with irregularly spaced points (“*lifting scheme*”: one can define wavelets adapted to the sample points), non gaussian noise;
- Jump detection;
- *Wavelet packets*.

State Space Modeling in Macroeconomics and Finance using S+FinMetrics

E. Zivot et al. (2002)

This is the documentation of the SsfPack part of S+FinMetrics, containing a lot of examples (Recursive Least Squares (RLS), CUSUM test (the cumulative sum of the residuals should have zero mean and a variance proportional to time; we can plot those cumulative sums of residuals and the confidence intervals), ARMA), with code – but it is rather useless if you do not have S-Plus and S+FinMetrics.

Searching for alpha, the importance of revenues and cash flow estimate revisions in stock selection

J. Gaudaut et al. (Citigroup, 2004)

Analysts can provide us with a few estimations for each stock:

- estimated earnings (very commonly used);
- estimated revenue;
- estimated cash flow.

The changes in those estimates (“revisions”) can help us select outperforming stocks. Actually, it suffices to look at the *breadth*, *i.e.*, the number of revisions. This strategy works better with the cash flow, continental Europe and low-capitalization values, but is sector-dependant.

New accounting standards changes are likely to increase the earnings estimates volatility and hence its usefulness; the other estimates may help.

Option Pricing under Model and Parameter Uncertainty using Predictive Densities

F.O. Bunnin et al. (2000)

This article (which I have not read) gives another example of Bayesian Model Averaging (BAM).

The theoretical price of an option is the expectation of its discounted payoff. To perform this computation, we assume that:

- We know the model describing the price of the underlying (usually the Black–Scholes model, which is known to be wrong);
- We know that the model parameters are constant (this is precisely the reason why the Black–Scholes model is wrong: the parameters are not constant (“clustered volatility”));
- We know those parameters (while different estimations of the volatility, such as the implied or the historical volatility, give different results).

With BAM, we can simultaneously consider several models (with constant or stochastic volatility) and take into account the uncertainty on the parameters; as a result, we get a distribution of theoretical prices.

Bayesian Model Averaging: A Tutorial **J.A. Hoeting et al. (1999)**

This article is a more comprehensive, detailed presentation of Bayesian Model Averaging (BMA) and its algorithms.

Here are a few means of averaging over the set of all models:

- Do not consider all the models but just the “best” and those close to it (a slightly different variable selection, a different error structure, etc.).
- *Occam’s window* suggests not to consider all the models but only the parsimonious and data-supported ones, *i.e.*, the models $M \in \mathcal{M}$ such that

- $\text{pr}(M|\text{Obs}) > (1 - \varepsilon) \max_{N \in \mathcal{M}} \text{pr}(N|\text{Obs})$
- there is no smaller model $N \subsetneq M$ that better fits the data, *i.e.*, such that $\text{pr}(N|\text{Obs}) > \text{pr}(M|\text{Obs})$.

Those models are enumerated by a Branch and Bound (B&B) algorithm (for small, simple problems).

- For larger problems, one can use the Markov Chain Monte Carlo (MCMC) algorithm to sample from the set of all models – this is called *Markov Chain Monte Carlo Model Composition* (MCMCMC or MC³).
- Orthogonalized model mixing (?).

The integrals (the posterior probability of the models) can be computed by

- explicit formulas for some (simple) classes of models;
- approximations (Laplace method);
- replacing $\text{pr}(y|M, \text{Obs})$ by $\text{pr}(y|M, \hat{\theta}_{\text{MLE}}, \text{Obs})$.

One may remark that these ideas allow us to compare completely unrelated models. If a model is embedded in the other, we can use a Likelihood Ratio (LR) test or the AIC, if not we can use the BIC approximation. Here, we have other approximations (actually, the Laplace approximation often coincides with the BIC one).

For linear regression, the choice of a model comprises:

- choice of variables;
- transformations: Box–Cox, ACE (Alternating Conditionnal Expectation: a family of non parametric transformations), broken lines (ACE often yields broken lines);
- outliers, using a model of the form $y = bx + \varepsilon$ where ε is a mixture of $N(0, \sigma^2)$ with probability $1 - p$ and $N(0, K^2 \sigma^2)$ with probability p , with p the proportion of outliers and $K \gg 1$ the variance-inflation parameter.

They also detail survival analysis and graphical models.

But how do we fix the prior model probabilities? One can set

$$\text{pr}(M) = \prod_j \pi^{\delta_j} (1 - \pi)^{1 - \delta_j}$$

where π is the probability of including a variable (with $\pi < \frac{1}{2}$, we put a penalty on larger models) and $\delta \in \{0, 1\}$ denotes the presence or absence of variable j in model M . Another solution is to ask an expert of the domain to forge “imaginary data”.

There are other approaches to model averaging:

- Bootstrap both the data and the model selection (the results are poorer than those of BMA);
- “Minimax Multiple Shrinkage Stein Estimator” (?);
- Bagging;
- Stacking (?);
- Boosting (weighted bagging).

The article concludes with two very detailed real-life examples and is followed by a few comments.

**Methodology for Bayesian Model Averaging:
an Update
J.A. Hoeting (2002)**

When performing a regression (or any other statistical analysis), we are confronted to the choice of a model (e.g., the choice of a few variables to use as predictive variables, among dozens of variables). Oftentimes, the data do not suggest a *single* best model. To overcome that *model uncertainty*, one can do the computation (e.g., we may want to compute a forecast) for each model, compute the “probability of each model given the data”, and average over the models.

More formally, let

$M \in \mathcal{M}$: the models

Obs : the observed data

y : the parameter to be estimated,

or the variable to be predicted

$\text{pr}(M)$: prior probability of model M

$\text{pr}(\theta|M)$: prior distribution of parameter θ in model M

$$\text{pr}(y|\text{Obs}) = \sum_{M \in \mathcal{M}} \text{pr}(y|\text{Obs}, M) \cdot \text{pr}(M|\text{Obs}) \quad (1)$$

$$\text{pr}(M|\text{Obs}) = \frac{\text{pr}(\text{Obs}|M) \cdot \text{pr}(M)}{\sum_{N \in \mathcal{M}} \text{pr}(\text{Obs}|N) \cdot \text{pr}(N)} \quad (2)$$

$$\text{pr}(\text{Obs}|M) = \int_{\theta \in \Theta} \text{pr}(\text{Obs}|M, \theta) d\theta. \quad (3)$$

To use such an approach, one has to overcome some problems:

- How to choose the prior probabilities?
- How to compute the integrals?
- How to deal with the large number of models?

For the first point, I have no clue; for the second, one uses ad hoc, model-dependent approximations; for the third, one uses Monte Carlo Markov Chains (MCMC) to sample in the set of models.

BMA can be applied, for instance, to the following sets of models:

- Linear models, with a different choice of variables, a different choice of transformations, a different selection of outliers;
- Non-parametric regression models;
- Spacial models (but I do not know anything about spacial models...);
- Generalized Linear Models (GLM);
- Graphical models;
- Classification trees (this is very similar to *bagging*);

- Survival analysis.

The article mentions a few S-Plus packages to perform BMA (apparently, they do not work under R in a straightforward way (they use undocumented internal S-Plus functions) and they have not been ported to R); a more comprehensive list is on the BMA home page, <http://www.research.att.com/~volinsky/bma.html>.

To sum up, the advantages of BMA are:

- Takes model uncertainty into account (if there is no model uncertainty, it is useless; it is more useful when there are many variables, many useless variables and multicollinearity problems);
- Better forecasts;
- Less over-optimistic confidence intervals (more classical model selection procedures use the same data twice: first to select the model, then to estimate the quantities of interest).

**Time Series in Finance:
the array database approach
D. Shasha**

A time series database should have the following desirable properties:

- Handling of regular time series, with missing values;
- Converting a time series to a given frequency, taking its *type* into account (for *level* time series, just take the latest value; for *flow* time series, take the sum since the last timestamp), in both directions (lower frequency: sampling, higher frequency: interpolation) – this also includes turning an irregular time series into a regular one;
- Perform basic computations (cumulative sum, moving average, correlation, regression, acf, etc.) and user-defined ones;
- Sequences are first-class objects, *i.e.*, you can play with them as easily as if they were numbers;
- *Bitemporality*: each event has two timestamps, the time at which it is valid and the time at which it was entered into the database; in other words, the database keeps track of how the errors were corrected. Thus, when backtesting (*i.e.*, when doing tests with historical data), we can use either the “historical data we should have had at the time, had the database been error-free and up-to-date” or the “historical data we actually had at the time, including the errors and the delays” – the former could in fact assume that we had the information before everyone else, which is not very realistic...

They detail several systems: FAME (Forecasting, Analysis and ModEling), S-Plus, SAS and KSQL (K is an array language – it makes me think of stored procedures in R under PostgreSQL).

The document also presents many concrete and detailed use cases; it ends with a large (but perhaps a bit outdated) bibliography on sequence similarity, subsequence matching, pattern matching.

Evanesce Implementation in S-Plus FinMetrics Module Insightful Corp., 2002

Copulas are a mean of describing the dependance relation between two random variables (X, Y) abstracting the peculiarities of the distribution of X and Y – we only focus on the dependance relation. The idea is to transform X and Y so that their distribution be uniform in $[0, 1]$. A *copula* is the joint distribution of the transformed variables.

This document lists more than a dozen copulas. Let us mention the simplest ones.

- Gaussian copulas (obtained when (X, Y) is gaussian, with correlation ρ);
- Mixture-of-gaussians copulas;
- Extreme value copulas: this is a *max-stable* family of copula; they verify

$$\forall u, v \quad C(u^t, v^t) = C(u, v)^t;$$

the Gumbel copula

$$C(u, v) = \exp - ((-\log u)^\delta + (-\log v)^\delta)^{1/\delta}$$

is one of them;

- Archimedean copulas,

$$C(u, v) = \phi^{-1}(\phi(u) + \phi(v))$$

for some function ϕ – this reduces the study to functions of a single variable. The Gumbel copula is archimedean.

They only present the notion of copulas for *pairs* of variables: actually, it can be generalized in higher dimensions. This is especially easy for families of copulas defined by a function of a single variable such as the archimedean copulas:

$$C(u_1, \dots, u_n) = \phi^{-1}(\phi(u_1) + \dots + \phi(u_n)).$$

You might also want to read *Taken to the limit: simple and not so simple loan loss distribution*, P.G. Schönbucher, http://www.wilmott.com/pdfs/040404_schon.pdf.

Orla, a data flow programming system for very large time series G. Zumbach and A. Trapletti (Olsen & Associates)

Flow programming languages (see *Advanced Programming Language Design*, by R. A. Finkel, for a description of the main families of programming languages and their characteristic features) are quite rare outside electronics. The authors have developed a framework based on the flow programming paradigm to process time series. Their data is stored in a “time series data base” (aka *temporal database*) that can be queried in a very simple way, close to:

```
SELECT FX(USD,*)
BETWEEN 2004-01-01 AND now;
```

(actually, it does not look like SQL, but it is as straightforward – they call that syntax SQDADL (Sequencial Data Description Language).

Iterative Incremental Clustering of Time Series J. Lin et al.

Clustering algorithms such as *k*-means (that tries to find spherical clusters) or EM (that tries to find elliptical clusters – more precisely, it is a model-based clustering algorithm that assumes the data is a mixture of (say) gaussians – in R, it is implemented in the *mclust* module) require a *good* first estimate of the cluster centers, that is progressively refined.

Instead of considering the whole data set right away (it may have a lot of variables), one may perform a clustering on the first coefficients of the Haar wavelet transform; then on the coefficients of levels 1 and 2, using the preceding results as initial estimates; etc.

The results are better than with a “batch” algorithm (we are less likely to find in a local extremum) – and it is faster.

To vary or not to vary? Weights, breaks, factors and sampling frequencies in predictive return models R. Masih (Goldman Sachs, 2004)

Factors vary with time. They suggest two ways of accommodating those changes.

First, you can take the same factor at several frequencies (daily, weekly, monthly) and use all of them in the same regression (MIDAS: MIXed Data Sampling).

Second, you can take several simple models (linear regression with a single predictive variable, for all the possible variables) and combine the models, with “bayesian methods” (they are very vague about those).

They also speak of “breaks”, but I know neither how they find them nor how they use them.

**Quantitative Portfolio Strategy
Peculiarity of the Japanese Stock Market
T. Suwabe (Goldman Sachs, 2004)**

Investor sentiment, defined as

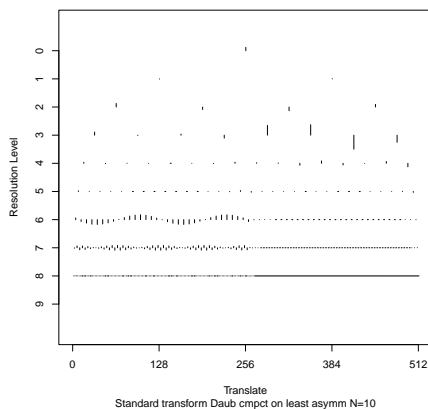
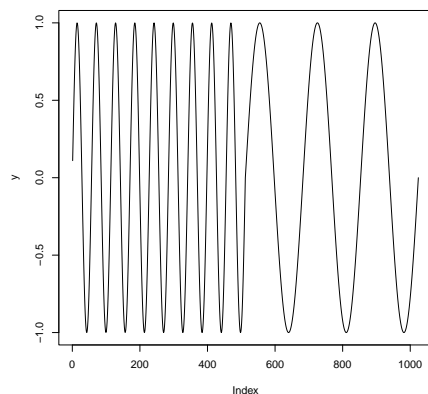
$$\text{NIS} = \frac{\text{Buy volume} - \text{Sell volume}}{\text{Total volume}},$$

estimated with intraday data, is correlated with future return.

Incidentally, they also define the notion of “factor return” (the coefficients of a factor, in a regression, when you do the same regression at various times); remark that they are autocorrelated in Europe and the US but not in Japan or in Asia; remark that Japanese returns have a negative autocorrelation that tends quickly to zero while the NIS has positive autocorrelation and tends slowly to zero.

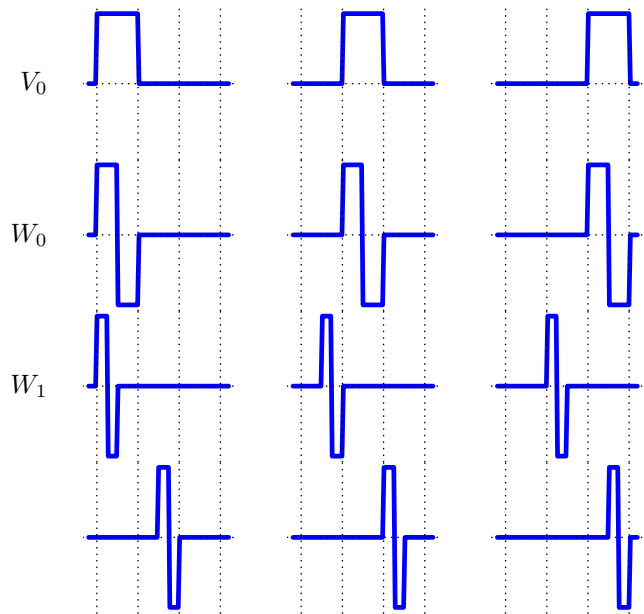
**Wavelets, Approximation and
Statistical Applications
W. Härdle et al., 1997**

The following figure represents a wavelet transform: the curve in the first plot is the signal to be described; the rectangles (or vertical segments) represent the coefficients; the lower left part tells us that in the left part of the interval, high frequencies dominate; the top right part tells us that in the right part of the interval, low frequencies dominate.



This is very similar to the decomposition of periodic functions into Fourier series, but we have one more dimension: the frequency spectrum is replaced by a frequency-location plot (an other advantage is that there is almost no Gibbs effect – very convenient when the signals are not periodic or stationary...)

The general idea is the following: find an orthonormal basis of L^2 whose elements can be interpreted in terms of “location” and “frequency”. For example (this is called the Haar basis):



This gives a decomposition

$$L^2 = V_0 \oplus \bigoplus_j W_j$$

where V_0 is generated by the father wavelet ϕ and its translations and the W_j are generated by the mother wavelet ψ , its translations and rescalings.

$$\begin{aligned} \phi &= \text{father wavelet} \\ \psi &= \text{mother wavelet} \end{aligned}$$

The choice of the basis depends on the applications: will you be dealing with smooth (\mathcal{C}^∞) functions? continuous but non-differentiable functions, with a fractal nature? functions with jumps? The convergence speed of the Wavelet decomposition will depend on the adequacy of the basis and the data studied.

More technically, the construction of a basis goes as follows.

1. Choose a father wavelet ϕ .
2. Set $\phi_j k(x) = 2^{j/2} \phi(2^j x - k)$, for $j \in \mathbf{N}$ and $k \in \mathbf{Z}$ – this is f translated by k and shrunk j times.
3. Set $V_j = \text{Vect} \{ \phi_{jk}, k \in \mathbf{Z} \}$ and check that:

4. $(\phi_{0k})_{k \in \mathbf{Z}}$ is an orthonormal system (a Hilbert basis of V_0 ;
5. $\forall j \in \mathbf{Z} \quad V_j \subset V_{j+1}$;
6. $\overline{\bigcup_{j \geq 0} V_j} = L^2$.
7. Then, set $W_j = V_{j+1} \ominus V_j$ so that $L^2 = V_0 \oplus \bigoplus_{j \geq 0} W_j$.
We then want to find a mother wavelet $\psi \in W_0$ such that if we set
8. $\psi_{jk}(x) = 2^{j/2} \psi(2^j x - k)$,
9. then $(\psi_{jk})_{k \in \mathbf{Z}}$ is a basis of W_j .
10. Then (finally), we have a basis of L^2 :

$$\begin{array}{ccccccc}
 \cdots & \phi_{0,-2} & \phi_{0,-1} & \phi_{0,0} & \phi_{0,1} & \phi_{0,2} & \cdots \\
 \cdots & \phi_{1,-2} & \phi_{1,-1} & \phi_{1,0} & \phi_{1,1} & \phi_{1,2} & \cdots \\
 \cdots & \phi_{2,-2} & \phi_{2,-1} & \phi_{2,0} & \phi_{2,1} & \phi_{2,2} & \cdots \\
 & & & \vdots & & &
 \end{array}$$

When conditions 4, 5 and 6 are satisfied, we say that we have found a MultiResolution Analysis (MRA).

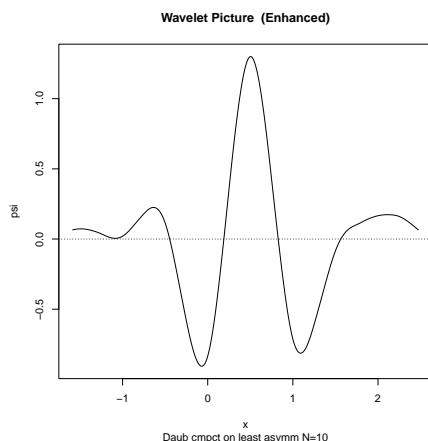
For the forgetful reader, they recall some basic facts about Fourier Analysis – if we just want to *use* wavelets, we do not need it: it is used to check that the abovementioned properties are satisfied, when you build your own wavelet basis.

The book then gives a first construction of wavelet bases, from a Riesz basis (it is not that hard, but we do not really need to know what it is): we get the Battle–Lemarié father wavelets.

Apart from the first one (the Haar father wavelet), they are not compactly supported.

The book then goes on to the Daubechies's construction of wavelet bases (based on Fourier analysis). This time:

- father and mother wavelets have a compact support;
- the first moment of the father wavelet are vanishing.



Let us list the main (families of) wavelet bases mentioned in the book:

- Haar;
- Battle–Lemarié;
- Daubechies;
- Coiflets (the father wavelet has vanishing moments);
- Symlets (more symmetric than the Daubechies wavelets).

(There are two technical chapters on Sobolev and Besov spaces, which I have not read.)

Let us now mention some statistical applications of wavelets:

- Approximation of possibly irregular functions and surfaces (there is a reduced Gibbs phenomenon, but you can circumvent it by using *translation-invariant wavelets*).
- Smoothing.
- Density estimation: just try to estimate the density as a linear combination of wavelets, then remove/threshold the wavelets whose coefficients are too small (usually, the threshold is chosen as 0.4 to 0.8 times the maximum coefficient; we also use *block thresholding*: we do not remove single coefficients but whole blocks of coefficients). For instance, density estimation of financial returns: we can clearly see the fat tails. Kerlel methods do not perform that well, unless they use adaptive bandwidth.
- Regression: this is very similar, we try to find a function f (linear combination of wavelets, we threshold the smaller coefficients) such that

$$\forall i \quad y_i = f(x_i) + \text{noise}.$$

If $x_i = i/n$ with $n = 2^k$ for some $k \in \mathbf{Z}$, it works out of the box otherwise, you have to scale and bin the data.

- Gaussian White Noise Estimation: find a function f such that

$$dY(t) = f(t) dt + \varepsilon dW(t) \quad t \in [0, 1].$$

- Jump detection.
- Time series.
- Diffusion Models (?).
- Image compression, indexing, reconstruction, etc.

Problems:

- The authors do not seem to know the symbols \forall and \exists – they systematically omit them, hoping that the reader will pick the right one;
- The book is rather old and therefore does not mention recent developments – it is good as an introduction to the subject, but should be complemented by other documents.

For more about wavelets, check <http://www.wavelet.org/> and its tutorials.

Computational Methods for Time Series **G. Kitagawa, T. Higuchi, S. Sato**

Linear State Space Models (SSM) are used to model non-stationnary time series; but when the series have jumps, outliers or involve non-gaussian distributions, we can turn to *non-linear state space models*:

$$x_n \sim Q_n(\cdot|x_{n-1})$$

$$y_n \sim R_n(\cdot|x_n).$$

The extended Kalman filter used in this context is only an approximation (a Taylor series, in fact) and it still assumes that the distributions are gaussian. If we do not want this approximation or if our distributions are really non gaussian, we can use the exact formulas:

$$p(x_n|y_{n-1}) = \int p(x_n|x_{n-1})p(x_{n-1}|y_{n-1}) dx_{n-1}$$

$$p(x_n|y_n) = \frac{p(y_n|x_n)p(x_n|y_{n-1})}{\int p(y_n|x_n)p(x_n|y_{n-1})}.$$

We have integrals when we go from n to $n+1$, but actually, we want to go from 1 to $N+1$ – we have integrals of integrals of... of integrals: all in all, we have to integrate over a high-dimensional space. We resort to *Monte Carlo integration*. This is called a *Particle Kalman Filter* (the idea is similar to that of using Monte Carlo simulations to price exotic options).

We usually estimate the model parameters by Maximum Likelihood (MLE), but this assumes that those parameters are constant – often, this is not a reasonable assumption. To solve this problem, we can augment the model by adding the parameters to the list of hidden variables: this is called a *Self-Organizing SSM*.

The articles also explains how the computations can be parallelized, how the particles can be exchanged between the different runs (*cross-over*).

Pattern Recognition of Time Series Using Wavelets **E. A. Maharaj**

The wavelets coefficients of time series can be used to estimate the “difference” (the distance) between two time series. One can devise a test to check if the coefficients of two series are significantly different: compute

$$\frac{(\text{wavelet coefficients of the first series})^2}{(\text{coefficients of the other})^2}$$

and compare it with the distribution obtained by randomly swapping the coefficients of the two series (they did the test with 256 and 1024 coefficients). One could use the resulting p -value to compute a distance between time series.

Personnal remarks:

- One might use that to *cluster* time series (into sectors, industries, countries, etc. – the interpretation will be left to more economically-inclined people).
- The coefficients may be similar but with delays (one could replace the distance between x and y by the minimum distance between delayed versions of x and delayed versions of y).
- Those delays can vary with time (a few days in normal market situations, much less in a crash – here, I should tell more about time warp distance and dynamic programming – the idea is the following: compute a time-warp distance for each level of the wavelets decomposition, then combine these distances).

Random and Changing Coefficient Models **G.C. Chow, Handook of Econometrics (1984),** **volume 2, chapter 21,**

A survey article, with too many formulas to be enlightening. A section presents various tests on those models (such as “are the coefficients random or constant?”).

The Kalman Foundations of Adaptive Least Squares **J.H. McCulloch, 2004**

The model underlying a regression looks like

$$y = \beta x + \text{noise}.$$

But in a financial context, the observations are ordered $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ and the coefficient β may (slowly) change with time. The model is then

$$\beta = \text{random walk}$$

$$y_t = \beta_t x_t + \text{noise}.$$

This is a State Space Model, but a non-linear one.

$\beta_{t+1} = \beta_t + \text{noise}$	(hidden)
$x_t = \text{noise}$	(observed)
$y_t = \beta_t x_t + \text{noise}$	(observed)

It is called a *Random Coefficient Model* (there are variants where β is stationnary) and the corresponding non-linear Kalman filter is called *Adaptive Least Squares* (ALS). This is a generalization of the Recursive Least Squares alluded to above.

The article is full of formulas and thus not very readable.

Here are some simpler alternatives to ALS:

- *Local Regression*, *i.e.*, for each point we do a weighted regression with a null weight for future observations (because we do not know them yet) and an exponentially decreasing weight for past observations; the difference with ALS is that we have to choose the rate of decrease of the weights.
- A broken-line regression (it sounds appealing, especially if we look at real data, but unfortunately it is not robust at all).

***What Investors Can Learn
from a Very Alternative Market***

**R.N. Kahn, Financial Analysts Journal 60 5,
p. 17–21.**

The article stresses the “importance of rigorous scientific analysis to successful investing”.

“Investors (or baseball team managers) are irrational in systematic and predictable ways:

- social interaction (herding);
- heuristic simplification (generalization from personal experience, extrapolating from recent events);
- self-deception (attributing positive outcomes to one’s skill and negative outcomes to bad luck).”

The low liquidity of the baseball market makes the resulting inefficiencies easier to spot and exploit.

(These ideas are presented and developed in the book *Moneyball*, by M. Lewis (2003).)

***Statistical Algorithms for Models
in State Space using SsfPack 2.2***

**S.J. Koopman, N. Shephard, J.A. Doornik
Econometrics Journal (1999) 2, p. 113–166**

This article presents SsfPack, an Ox library for State Space Model estimations (Ox is a C-like language for matrix and time-series analysis, commercial, closed-source and Windows-only – there is a limited Linux version, available only to academic users). The SPlus module “FinMetrics” uses SsfPack.

The article contains a lot of examples and has a distinctive bayesian flavor.

The first part presents various time series models that may be expressed in state-space form: ARMA, structured model with seasonal and *cycle* components

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} + \text{noise},$$

regression models (*recursive least squares* – we want to find β such that $y_t = \beta x_t + \text{noise}$: take a first, random, estimation $\hat{\alpha}_0$ of β , consider the first observation (x_1, y_1) , refine the estimate $\hat{\alpha}_1$, consider the second observation, etc.), non-parametric cubic spline models (not read).

The second part presents the implemented algorithms: simulation, Kalman filter, smoother (in a filter, you only use the information available up to time t to predict the hidden variable at time t , in a smoother, you use all the information, even that posterior to the prediction), a posteriori sampler (I think – it was not that clear), likelihood computation

$$-\frac{nN}{2} \log(2\pi) - \frac{1}{2} (\log |F_t| + v'_t F_t^{-1} v_t)$$

(where n is the number of observations, N the number of observed variables, $|\cdot|$ the determinant, v_t the residuals, F_t the variance of the residuals) – there is also a “concentrated or profile log-likelihood”.

***“We’re going to be a 90 % Linux shop by 2006”*
Linux Journal 127, November 2004, p. 48-50**

These words are from Morgan Stanley.

***Scientific Visialization with PoVRay*
Linux Journal 127, November 2004, p. 62-67**

PoVRay is a ray tracer, *i.e.*, a program to draw 3D scenes from a textual description. It can draw isosurfaces, *i.e.*, surfaces of the form $f(x, y, z) = k$, where f is specified by an equation or a binary df3 file (8-bit by default, but a patch extends this to 32-bit data and another patch to HDF files, that contain several functions). It is being used for meteorogological data.

***Maximum likelihood estimation of
mean-reverting processes*
J.C.G. Franco**

MLE of a mean-reverting Ornstein-Uhlenbeck process (OU is a continuous analogue of an AR(1) time series, the MR version is used to model exchange rates or the price of products with a fixed cost of production) requires either maximizing a function of 3 variables or solving a (non-linear) system of 3 equations. This article mixes these two approaches and reduces the problem to that of maximizing a function of a single variable.

***Swingtum – A Computational Theory of
Fractal Dynamic Swings and Physical Cycles
of Stock Market in a Quantum Price-Time
Space*
H. Pan**

The market reacts to 4 types of fluctuations:

- dynamic swings (they have a fractal nature, have multilevel trends, as Elliott waves for instance);
- physical cycles (year, month, week);
- abrupt momentum (news impact, chart patterns);
- random walk (the rest, what we cannot predict);

Nothing really concrete:

- If the prices depend on several “forces”, then the price variation is the “sum” of these “forces” (he takes into account a stock “mass” , *i.e.*, inertia);
- The prices are piecewise log-periodic,

$$p \sim \sin \log t,$$

i.e., the rise is steeper than the fall (or conversely) and the waves start and end at random instants;

- Hilbert transforms are important (I have no idea why).
- “Multidimensionnal embedding” (yes, these words are mentioned without any further explanation).

Yet, if we remove all the nonsense from this gibberish, the article suggests to do exactly what we are doing: look at the data, all the data, cluster it (empirically or not, your clustering may/should be hierarchical) and try to find a model (using the rational and irrational beliefs investors are actually using, because their actions depend on that and the market responds to those actions).

***Bubbles, Crashes and Intermittency
in Agent-Based Market Models***
I. Giardina and J.-P. Bouchaud
arXiv:cond-mat/0206222

A very readable article on the same subject. They present a simplified model of financial markets:

- There are only two assets, a stock and a bond (risk-free asset).
- Each agent has a predefined set of strategies. He looks at the last m returns, compares them with the risk-free returns, obtains a history binary vector of length m ; a strategy is a function $\{0,1\}^m \rightarrow \{\text{buy}, \text{sell}\}$ (chartists do that). The strategies are chosen at random with a bias, P (“polarization”) towards “trend-following” or “contrarian” strategies.
- There is also a “do nothing” strategy.
- If the prices become too high or too low, the agents revert to fundamental analysis.
- If an agent decides to invest, he only invests a fraction g of his assets.
- Each strategy is rated with the number of “successes” it would have yielded in the past (weighted moving average).
- The prices depend on the number of actors willing to buy/sell and on the market stiffness λ .

The consequences:

- The system has three states: oscillating (bubbles and crashes), intermittent (irregular bubbles and crashes) and stable.
- Only two parameters are important: polarization P and g/λ .
- There are multiscale fluctuations (even though the model is not multiscale).
- There is volatility clustering and long-term dependence.
- A Tobin tax of a few basis points would have no effect; A Tobin tax of a few percents have a stabilizing effect.

***Evolution Management in a Complex
Adaptive System***
D.M.D. Smith, 2003

Detailed version of the preceding article – still unreadable but less empty.

***Evolution Management in a Complex
Adaptive System: Engineering the Future***
D.M.D. Smith and N.F. Johnson
arXiv:cond-mat/0409036

The article alluded to by the New Scientist article: unreadable and empty.

Forecasting Stock Markets
New Scientist, 25th September 2004, p. 16

Presentation of agent-based models of stock markets: they have the same properties as the actual stock markets (not new); furthermore, one can easily *control* those markets (this is new). It could have applications beyond finance (meteorology, pollution, terraforming, etc.)

***A Joint Review of Technical and Quantitative
Analysis of Financial Markets Towards a
Unified Science of Intelligent Finance***
H. Pan

This article starts by a presentation of fundamental analysis, technical analysis and quantitative analysis of financial markets.

Fundamental analysis is the study of the accounting variables tied to a given asset: the sources are irregular, not always reliable, the data are published with a certain time lag, the accounting rules differ from country to country (which prevents comparison). Yet, automatic news monitoring could provide some insight (I would like to hear more about the applications of NLP (*Natural Language Processing*) in finance: is it really used?).

Technical analysis is the study of the price and volume times series and the quest of *signs* in those series.

The *Dow theory of trends* states that:

- There are three time scales (year, month, week);
- At each scale, the trends have 3 phases: accumulation (knowledgeable investors buy/sell), public participation (knowledgeable investors wait, the public buys/sells), distribution phase (knowledgeable investors sell/buy).

The *Elliott theory of waves* states that a trend (which can be observed at various scales) looks like that:

(insert a picture)

The ratios of the price differences are believed to be Fibonacci numbers or $k/2^n$ (this is an irrational belief).

The *Gann theory of cycles and angles* represents those waves in a 2-dimensional space, price×time: it can be analysed statically (one looks at the densest regions) or dynamically (given the current region, which is the next most probable region? – this is a Markov chain). Marketing people, undeterred by the perspective of falling in the “abuse of science” trap exemplified by the Sokal affair, see an analogy between “price-time” and “space-time”, and between the movements between the price-time space and the electrons jumping between energy levels – it may impress clients, though.

A technical analyst typically chooses a minimal set of technical indicators:

- A market-mode indicator, indicating if the market is in a trend or in a cycle, such as the “MESA filter” (but we do not know what it is);
- A trend indicator (a moving average);
- A cycle indicator, such as RSI or Sinewave (again, I do not know what it is);
- A volatility indicator, such as the 2σ Bollinger bands;
- A market breadth indicator (again, I do not understand).

A technical analyst also tries to spot *chart patterns*, at various time scales.

Quantitative finance has pointed out the following facts:

- The distribution of returns is not normal, it has fat tails, it sometimes looks *stable* (the Levy stable distributions are a family of distributions such that if X_1, \dots, X_n are iid and follow one of those, then so does their sum $X_1 + \dots + X_n$ – the most prominent examples are the normal and Cauchy distributions) (but this is contradicted by the facts that the second moment is finite, or the fact that the extreme value distribution follows a power law); the longer the time scale, the more normal the distribution looks;
- *Clustered volatility*, i.e., heteroscedasticity of returns, can be accounted for by GARCH models;

- Stock prices have a *fractal* structure;
- Crashes can be modelled as *phase transitions*;
- *Multiagent game models* such as the Minority Game or the Santa Fe Institute Stock Market Model can explain some of these characteristics of the distribution of returns.

The *Swing Market Hypothesis* states that the markets are sometimes efficient, sometimes not; in each of these modes, they can have several regimes. There are 4 types of fluctuations:

- dynamic swings (fractals, i.e., multilevel trends, i.e., Elliott waves, with “power laws” and “log-periodicity”);
- physical cycles (year, month, week);
- abrupt momentum (news impact, chart patterns);
- random walks.

***The good and the bad of value investing:
applying a bayesian approach
to develop enhancement models
R. Bird***

“Value investing” means finding stocks that are misvalued by the market: if they are overvalued, we sell them, if they are undervalued, we buy them. Various ratios try to identify them: the article chose the *book to market*, but any other would do.

The problem is that among those value stocks, most will remain over- or under-valued – they were not real value stocks, their price was that high or that low for good reasons, the variables/methods we used to select them were too coarse to identify those reasons.

One can use *logistic regression* with respect to various fundamental variables to spot them – they do not take all the available variables but only those (less than 30) that were deemed important, according to previous articles. But there are still too many variables to have a reliable model. A simple approach would be to try to select “the best” (or, at least, “a good”) model, both simple and giving good predictions.

Instead, as we are not interested in the interpretation of the model but mainly in its prediction, we can consider several good models and compute their “average forecast” – this is called *bagging*.

Here, they take *all* the models and use the MCMC algorithm to average them – but this requires that we affect a *score* to each model: the article does not tell us how it is computed (actually, they are “just” using *Bayesian Model Averaging*).

(Here, I should recall what “MCMC” means...)

The conclusions of the article are:

- the model changes with time;

- the model depends on the country (this is not surprising, because the accounting rules, and hence the variables, depend on the country).

***Profitability, Earnings and Book Value
in Equity Valuation: A Geometric View
and Empirical Evidence***
P. Chen and G. Zhang

The article compares the model presented in the previous article with actual data. In short,

$$\text{value} \sim \text{Book Value} * f(\text{earnings}),$$

where f is convex, increasing.

***Accounting Information, Capital Investment
Decisions and Equity Valuation:
Theory and empirical Implications***
G. Zhang

One may model the evolution of a company as follows:

$$\begin{array}{ll} \text{Cash flow} & c_t = \kappa_t s_{t-1} \\ \text{Operating Efficiency} & \kappa_t = \text{random walk} \\ \text{Assets} & s_t = \gamma s_{t-1} + i_t \\ \text{Stock depreciation} & \gamma \\ \text{Investment} & i_t \\ \text{Value} & V_t = \sum_{s>t} \text{PV } E[\text{net cash flow}] \\ & = \sum_{s>t} \text{PV } E[c_t - i_t] \end{array}$$

Here, i_t is not specified: it depends on *investment decisions*. To simplify the model, they limit the choices to:

- stop investing;

- invest such that the value of the stocks s_t remains constant;
- invest more (with a growth factor G).

The actual investment decision is then the one that maximizes the value. After playing with the formulas, it becomes

$$\begin{aligned} V_t &= \text{expected value if we maintain the activities} \\ &+ \text{price of the option to stop operations at } t+1 \\ &+ \text{price of the option to expand the operations} \end{aligned}$$

and later, a function of the book value and the earnings. The article examines the convexity of the formula:

- V is convex;
- For steady-state companies, the value only depends on the earnings, not on the book value.

The PB-ROE Valuation Model Revisited
J.W. Wilcox and T.K. Philips, 2004

The P/B ROE valuation model is just another *valuation model* (as the DDM, Discounted Dividend Model). In short:

$$\log(\text{P/B}) = \alpha + \beta \text{ ROE}.$$

They estimate α and β with a classical regression. The fact that The P/B is above (respectively under) its predictive value does not necessarily mean that it is overvalued (respectively undervalued): other factors could account for the discrepancy.