Machine Learning 2

E. Le Pennec

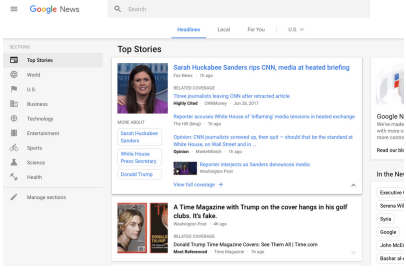
MAP541 - Machine Learning 2 - Spring 2023

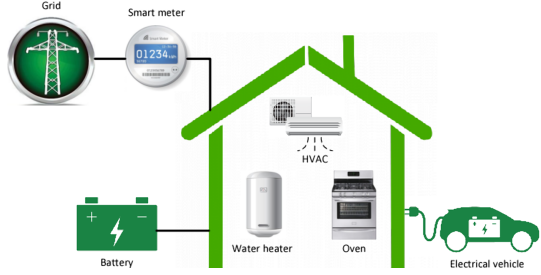
Outline

Outline

Outline

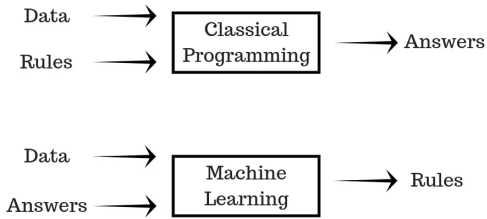
Machine Learning





Sources: MyCarDoesWhat.org/theverge.com/Zhiqiang Wan et al.

Machine Learning



A definition by Tom Mitchell

A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

Source: C. Barabasi

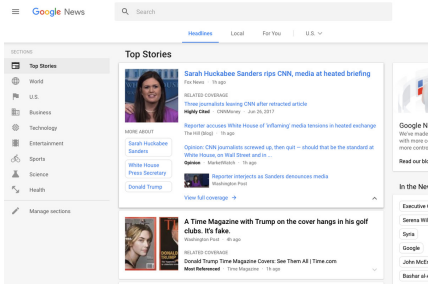
Object Detection

A detection algorithm:

- **Task**: say if an object is present or not in the image
- **Performance**: number of errors
- **Experience**: set of previously seen labeled images

Source: MyCarDoesWhat.org

Article Clustering

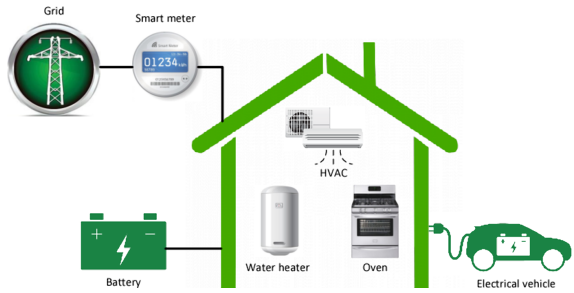


An article clustering algorithm:

- **Task**: group articles corresponding to the same news
- **Performance**: quality of the clusters
- **Experience**: set of articles

Source: theverge.com

Smart Grid Controler

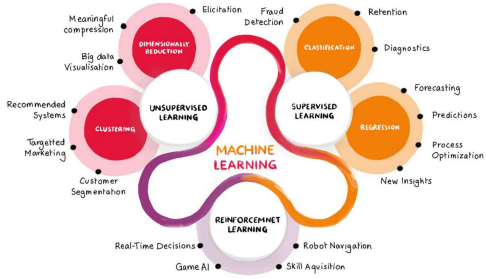


A controler in its sensors in a home smart grid:

- **Task**: control the devices
- **Performance**: energy costs
- **Experience**:
  - previous days
  - current environment and performed actions

Source: Zhiqiang Wan et al.

Three Kinds of Learning



Unsupervised Learning

- **Task:** Clustering/DR/Generative
- **Performance:** Quality
- **Experience:** Raw dataset (No Ground Truth)

Supervised Learning

- **Task:** Prediction/Classification
- **Performance:** Average error
- **Experience:** Good Predictions (Ground Truth)

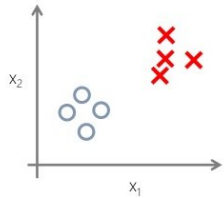
Reinforcement Learning

- **Task:** Actions
- **Performance:** Total reward
- **Experience:** Reward from env. (Interact. with env.)

Source: BCG

- **Timing:** Offline/Batch (learning from past data) vs Online (continuous learning)

Supervised and Unsupervised



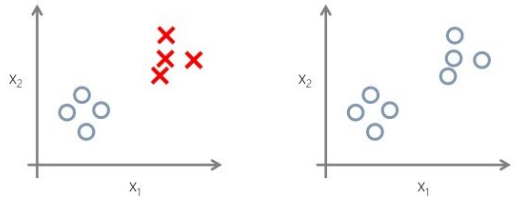
Supervised Learning (Imitation)

- **Goal:** Learn a function f predicting a variable Y from an individual $\underline{X}$.
- **Data:** Learning set with labeled examples $(\underline{X}_i, Y_i)$

- **Assumption:** Future data behaves as past data!
- **Predicting is not explaining!**

Source: KDnuggets

Supervised and Unsupervised



Supervised Learning (Imitation)

- **Goal:** Learn a function f predicting a variable Y from an individual $\underline{X}$.
- **Data:** Learning set with labeled examples $(\underline{X}_i, Y_i)$

- **Assumption:** Future data behaves as past data!
- **Predicting is not explaining!**

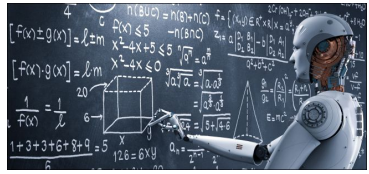
Unsupervised Learning (Structure Discovery)

- **Goal:** Discover a structure within a set of individuals $(\underline{X}_i)$.
- **Data:** Learning set with unlabeled examples $(\underline{X}_i)$

- Unsupervised learning is not a well-posed setting...

Source: KDnuggets

Machine Can and Cannot



Machine Can

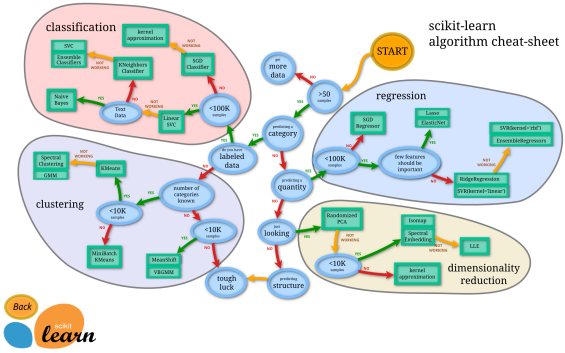
- Forecast (Prediction using the past)
- Detect expected changes
- Memorize/Reproduce
- Take a decision very quickly
- Learn from huge dataset
- Optimize a single task
- Replace/Help some humans

Machine Cannot

- Predict something never seen before
- Detect any new behaviour
- Create something brand new
- Understand the world
- Get smart really fast
- Go beyond their task
- Kill all humans

- Some progresses but still very far from the *singularity*...

Machine Learning

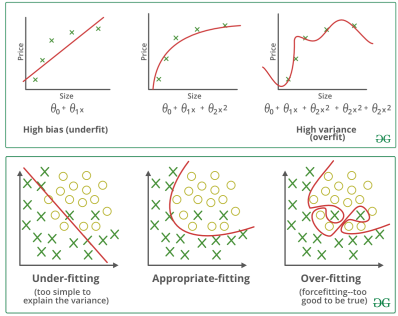


Machine Learning Methods

- Huge catalog of methods,
- Need to define the performance,
- Numerous tricks: feature design, hyperparameter selection...

Source: scikit-learn.org

Under and Over Fitting

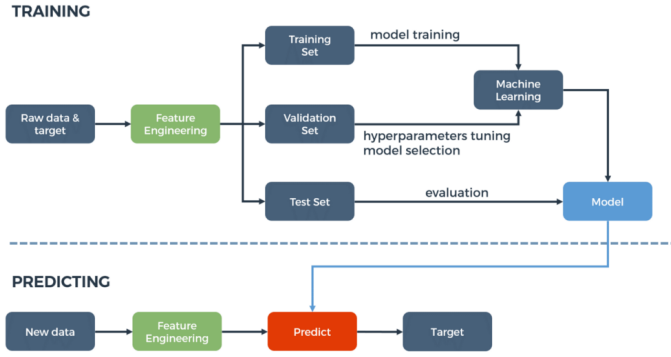


Finding the Right Complexity

- What is best?
  - A simple model that is stable but false? *(oversimplification)*
  - A very complex model that could be correct but is unstable? *(conspiracy theory)*
- Neither of them: tradeoff that depends on the dataset.

Source: geeksforgeeks.com

Machine Learning Pipeline

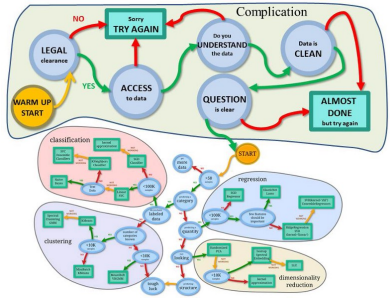


Learning pipeline

- Test and compare models.
- Deployment pipeline is different!

Source: CDiscount

Data Science $\neq$ Machine Learning

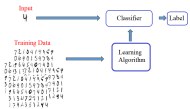


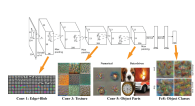
Main DS difficulties

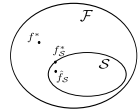
- Figuring out the problem,
- Formalizing it,
- Storing and accessing the data,
- Deploying the solution,
- Not (always) the Machine Learning part!

Source: Ch. Bourguignat

MAP 541 - ML 2 - Goal







Goal

- Complete your knowledge on classical supervised and non supervised method.
- Introduce you to recommender systems and reinforcement learning.
- Give you some basic idea on how to scale and deploy an algorithm.

Evaluation

- A practical lab (5 pt)
- A project (15 pt)

MAP 541 - Team

- Erwan Le Pennec

Erwan.Le-Pennec@polytechnique.edu

- Marine Le Morvan

marine.le-morvan@polytechnique.edu

- Edouard Oyallon

edouard.oyallon@cnrs.fr

- Kevin Scaman

kevin.scaman@gmail.com

MAP 541 - Schedule

7 Lectures (9h30-12h30)

- Thu. 19/01: Introduction, Error Estimation, Cross Validation and Auto ML
- Wed. 25/01: A Review of the Methods seen so far
- Wed. 01/02: Trees and Ensemble Methods
- Wed. 08/02: Unsupervised Learning: Beyond PCA and k-means
- Wed. 15/02: Recommender System and Matrix Factorization
- Wed. 22/02: Introduction to Reinforcement Learning
- Tue. 08/03: At Scale Machine Learning and Deployment

- Mon. 03/04: **Deadline for the project**

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

Outline

Supervised Learning

Supervised Learning Framework

- Input measurement $\underline{X} \in \mathcal{X}$
- Output measurement $Y \in \mathcal{Y}$.
- $(\underline{X}, Y) \sim \mathbb{P}$ with $\mathbb{P}$ unknown.
- **Training data** : $\mathcal{D}_n = \{(\underline{X}_1, Y_1), \ldots, (\underline{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbb{P}$)

- Often
  - $\underline{X} \in \mathbb{R}^d$ and $Y \in \{-1, 1\}$ (classification)
  - or $\underline{X} \in \mathbb{R}^d$ and $Y \in \mathbb{R}$ (regression).
- A **predictor** is a function in $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y} \text{ meas.}\}$

Goal

- Construct a **good** predictor $\widehat{f}$ from the training data.

- Need to specify the meaning of good.
- Classification and regression are almost the **same** problem!

Loss and Probabilistic Framework

Loss function for a generic predictor

- **Loss function**: $\ell(Y, f(\underline{X}))$ measures the goodness of the prediction of Y by $f(\underline{X})$
- Examples:
  - 0/1 loss: $\ell(Y, f(\underline{X})) = \mathbf{1}_{Y \neq f(\underline{X})}$
  - Quadratic loss: $\ell(Y, f(\underline{X})) = |Y - f(\underline{X})|^2$

Risk function

- Risk measured as the average loss for a new couple:

$$\mathcal{R}(f) = \mathbb{E}_{(\underline{X},Y)\sim\mathbb{P}}[\ell(Y, f(\underline{X}))]$$

- Examples:
  - 0/1 loss: $\mathbb{E}[\ell(Y, f(\underline{X}))] = \mathbb{P}(Y \neq f(\underline{X}))$
  - Quadratic loss: $\mathbb{E}[\ell(Y, f(\underline{X}))] = \mathbb{E}\left[|Y - f(\underline{X})|^2\right]$

- **Beware:** As $\widehat{f}$ depends on $\mathcal{D}_n$, $\mathcal{R}(\widehat{f})$ is a random variable!

Best Solution

- The best solution $f^\star$ (which is independent of $\mathcal{D}_n$) is

$$f^\star = \arg\min_{f \in \mathcal{F}} \mathcal{R}(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}[\ell(Y, f(\underline{X}))] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\Big[\mathbb{E}_{Y|\underline{X}}[\ell(Y, f(\underline{X}))]\Big]$$

Bayes Predictor (explicit solution)

- In binary classification with $0-1$ loss:

$$f^\star(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ & \Leftrightarrow \mathbb{P}(Y=+1|\underline{X}) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

- In regression with the quadratic loss

$$f^\star(\underline{X}) = \mathbb{E}[Y|\underline{X}]$$

Issue: Solution requires to **know** $\mathbb{E}[Y|\underline{X}]$ for all values of $\underline{X}$!

Goal

Machine Learning

- Learn a rule to construct a **predictor** $\widehat{f} \in \mathcal{F}$ from the training data $\mathcal{D}_n$ s.t. **the risk** $\mathcal{R}(\widehat{f})$ is **small on average** or with high probability with respect to $\mathcal{D}_n$.

- In practice, the rule should be an algorithm!

Canonical example: Empirical Risk Minimizer

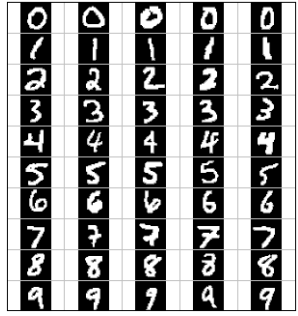
- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the empirical loss

$$\widehat{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(\underline{X}_i))$$

- Examples:
  - Linear regression
  - Linear classification with

$$\mathcal{S} = \{\underline{x} \mapsto \operatorname{sign}\{\underline{x}^\top \beta + \beta^{(0)}\} \, / \beta \in \mathbb{R}^d, \beta^{(0)} \in \mathbb{R}\}$$

Optical Character Recognition

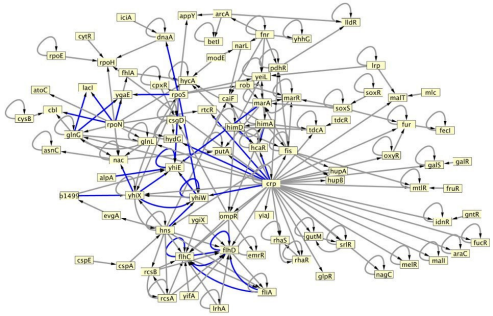


Reading a ZIP code on an envelop

- **Task:** give a number from an image.
- **Experience:** $\underline{X}$ = image / Y = corresponding number.
- **Performance measure:** error rate.

Source: Y. LeCun

Biology

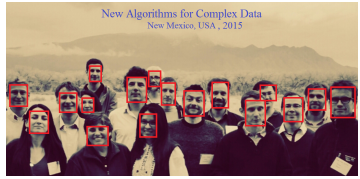


Predicting protein interaction

- **Task:** Predict (unknown) interactions between proteins.
- **Experience:** $\underline{X}$ = pair of proteins / Y = existence or no of interaction.
- **Performance measure:** error rate.
- Numerous similar questions in bio(informatics): genomic,...

Source: Unknown

Detection

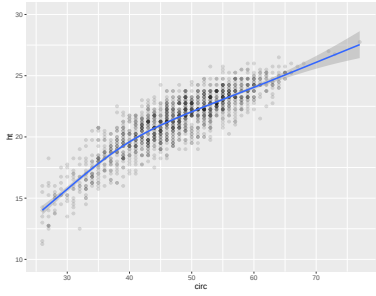


Face detection

- **Task:** Detect the position of faces in an image
- Different setting?
- Reformulation as a supervised learning problem.
- **Task:** Detect the presence of faces at several positions and scales.
- **Experience:** $\underline{X}$ = sub image / Y = presence or no of a face...
- **Performance measure:** error rate.
- Lots of detections in an image: post processing required...
- **Performance measure:** box precision.

Source: A. Fermin

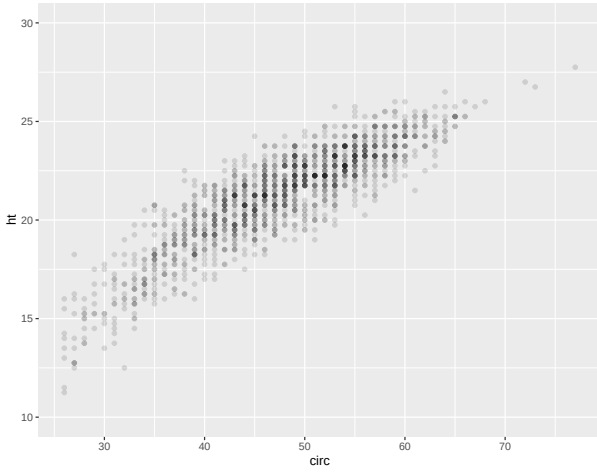
Eucalyptus



Height estimation

- Simple (and classical) dataset.
- **Task:** predict the height from circumference.
- **Experience:** $\underline{X}$ = circumference /
- Y = height.
- **Performance measure:** means squared error.

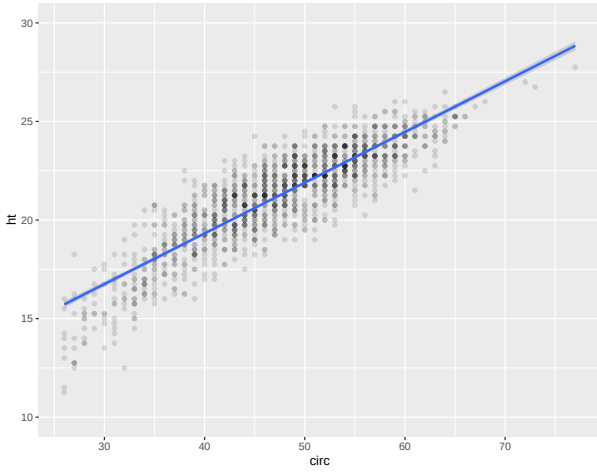
Eucalyptus



Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

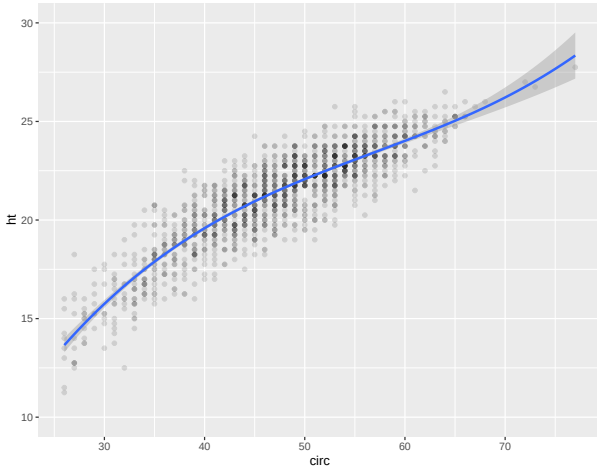
Eucalyptus



Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

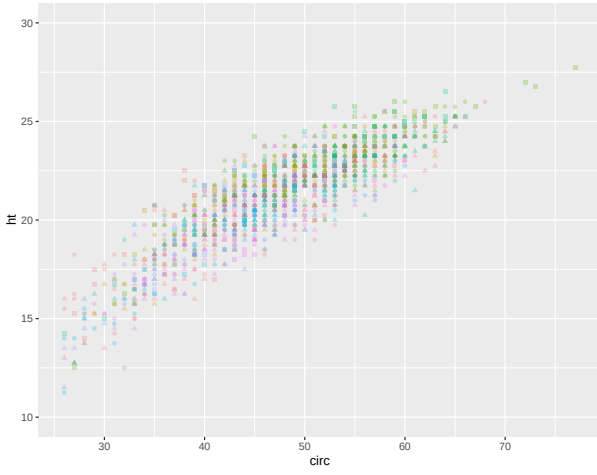
Eucalyptus



Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference / Y: height

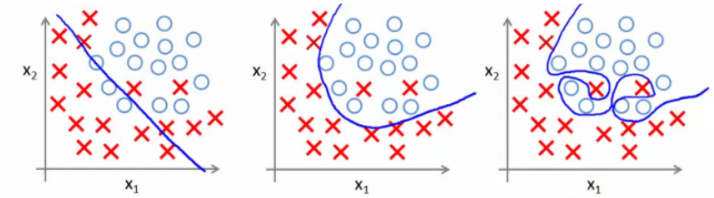
Eucalyptus



Dataset - P.A. Cornillon

- Real dataset of 1429 eucalyptus obtained by P.A. Cornillon:
  - $\underline{X}$: circumference, block, clone / Y: height

Under-fitting / Over-fitting Issue



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

(g = sigmoid function)

UNDERFITTING
(high bias)

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \ldots)$

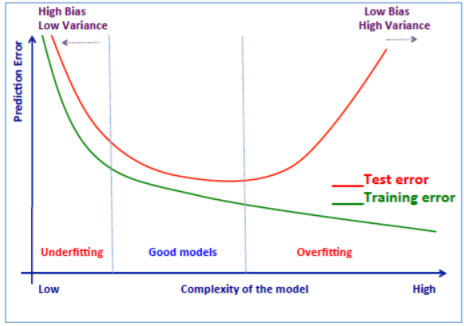
OVERFITTING
(high variance)

Model Complexity Dilemna

- What is best a simple or a complex model?
- Too simple to be good? Too complex to be learned?

Source: A. Ng

Under-fitting / Over-fitting Issue



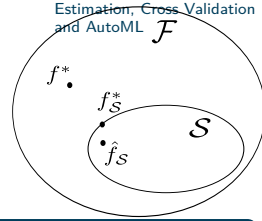
Under-fitting / Over-fitting

- **Under-fitting:** simple model are too simple.
- **Over-fitting:** complex model are too specific to the training set.

Source: Unknown

Bias-Variance Dilemma

- General setting:
  - $\mathcal{F} = \{\text{measurable functions } \mathcal{X} \to \mathcal{Y}\}$
  - Best solution: $f^\star = \text{argmin}_{f \in \mathcal{F}} \mathcal{R}(f)$
  - Class $\mathcal{S} \subset \mathcal{F}$ of functions
  - Ideal target in $\mathcal{S}$: $f^\star_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \mathcal{R}(f)$
  - Estimate in $\mathcal{S}$: $\widehat{f}_{\mathcal{S}}$ obtained with some procedure



Approximation error and estimation error (Bias/Variance)

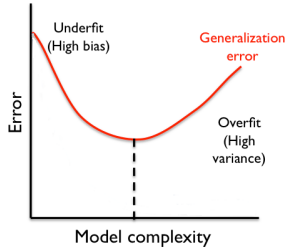
$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(f^\star_{\mathcal{S}}) - \mathcal{R}(f^\star)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star_{\mathcal{S}})}_{\text{Estimation error}}$$

- Approx. error can be large if the model $\mathcal{S}$ is not suitable.
- Estimation error can be large if the model is complex.

Agnostic approach

- No assumption (so far) on the law of $(\underline{X}, Y)$.

Under-fitting / Over-fitting Issue



- Different behavior for different model complexity
- **Low complexity model** are easily learned but the approximation error (**bias**) may be large (**Under-fit**).
- **High complexity model** may contain a good ideal target but the estimation error (**variance**) can be large (**Over-fit**)

Bias-variance trade-off $\iff$ avoid **overfitting** and **underfitting**

- **Rk:** Better to think in term of method (including feature engineering and specific algorithm) rather than only of model.

Source: Unknown

Theoretical Analysis

Statistical Learning Analysis

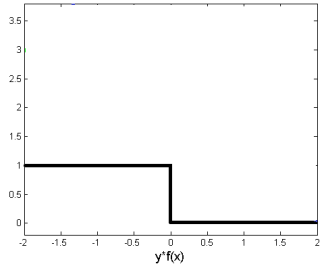
- Error decomposition:

$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star) = \underbrace{\mathcal{R}(f^\star_{\mathcal{S}}) - \mathcal{R}(f^\star)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^\star_{\mathcal{S}})}_{\text{Estimation error}}$$

- Bound on the approximation term: approximation theory.
- Probabilistic bound on the estimation term: probability theory!
- **Goal: Agnostic bounds**, i.e. bounds that do not require assumptions on $\mathbb{P}$! (Statistical Learning?)

- Often need mild assumptions on $\mathbb{P}$... (Nonparametric Statistics?)

Binary Classification Loss Issue

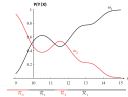


Empirical Risk Minimizer

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

- Classification loss: $\ell^{0/1}(y, f(\underline{x})) = \mathbf{1}_{y \neq f(\underline{x})}$
- Not convex and not smooth!

Probabilistic Point of View
Ideal Solution and Estimation



- The best solution $f^\star$ (which is independent of $\mathcal{D}_n$) is

$$f^\star = \arg\min_{f \in \mathcal{F}} \mathcal{R}(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}[\ell(Y, f(\underline{X}))] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\Big[\mathbb{E}_{Y|\underline{X}}[\ell(Y, f(\underline{x}))]\Big]$$

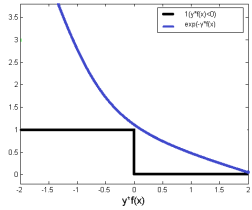
Bayes Predictor (explicit solution)

In binary classification with $0-1$ loss:

$$f^\star(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- **Issue:** Solution requires to **know** $\mathbb{E}[Y|\underline{X}]$ for all values of $\underline{X}$!
- **Solution:** Replace it by an estimate.

Optimization Point of View
Loss Convexification



Minimizer of the risk

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

- **Issue:** Classification loss is not convex or smooth.
- **Solution:** Replace it by a convex majorant.

Probabilistic and Optimization Framework

How to find a good function f with a *small* risk

$$\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\underline{X}))] \quad ?$$

Canonical approach: $\widehat{f}_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\underline{X}_i))$

Problems

- How to choose $\mathcal{S}$?
- How to compute the minimization?

A Probabilistic Point of View

Solution: For $\underline{X}$, estimate $Y|\underline{X}$ plug this estimate in the Bayes classifier:
(Generalized) Linear Models, Kernel methods, k-nn, Naive Bayes, Tree, Bagging...

An Optimization Point of View

Solution: If necessary replace the loss ℓ by an upper bound $\bar{\ell}$ and minimize the empirical loss: **SVR, SVM, Neural Network,Tree, Boosting...**

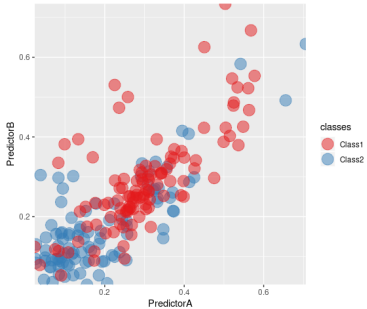
Outline

Example: TwoClass Dataset

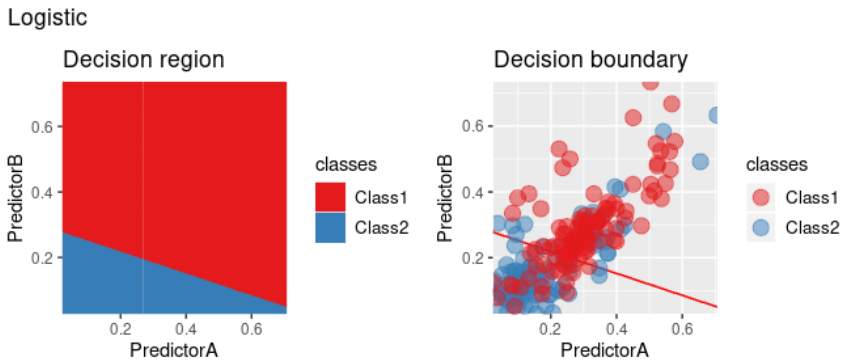
Synthetic Dataset

- Two features/covariates.
- Two classes.

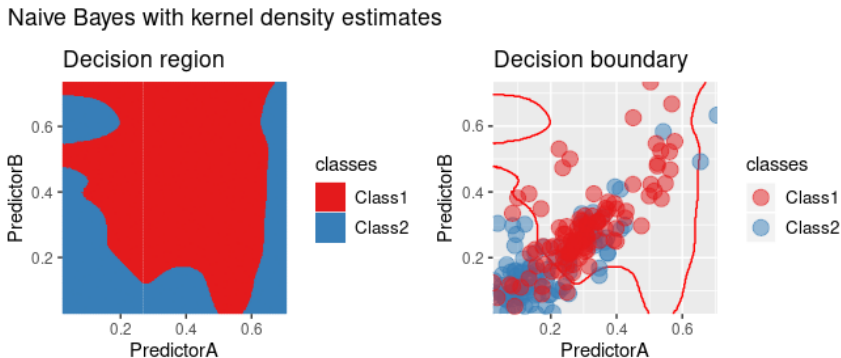
- Dataset from *Applied Predictive Modeling*, M. Kuhn and K. Johnson, Springer
- Numerical experiments with `R` and the {`caret`} package.



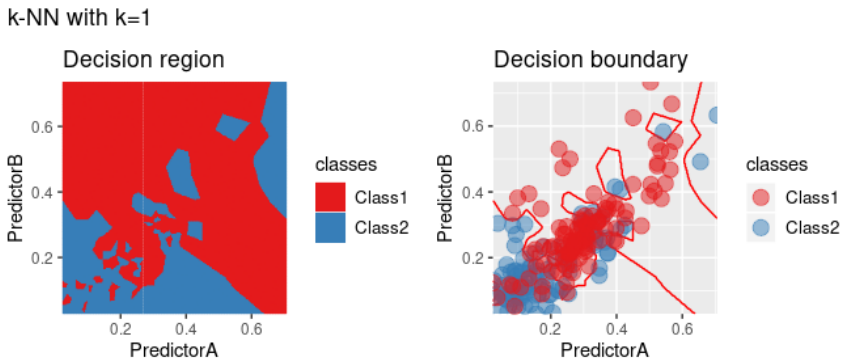
Example: Linear Discrimination



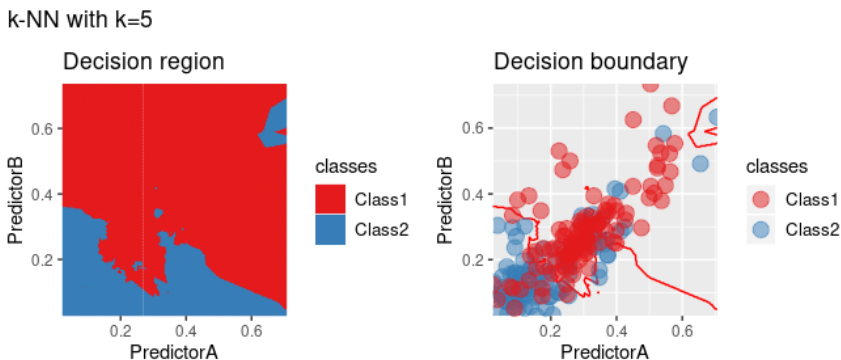
Example: More Complex Model



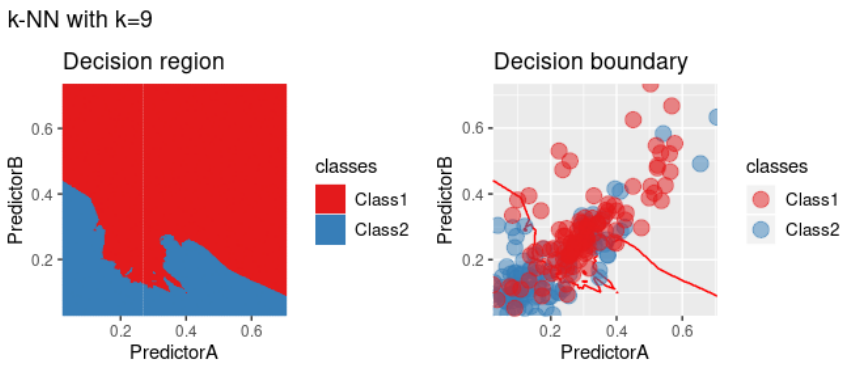
Example: KNN


k-NN with k=1
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

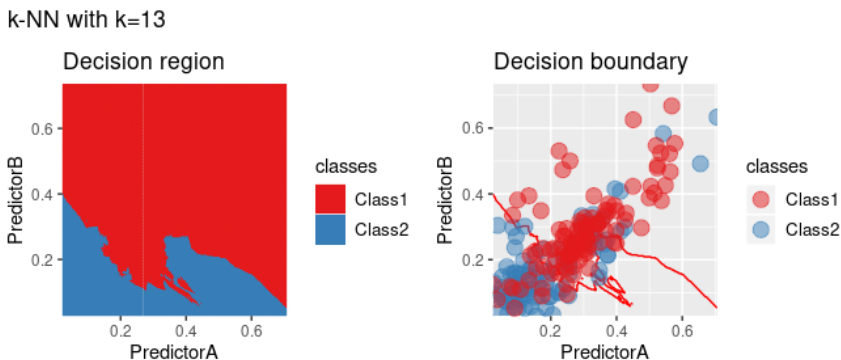
Example: KNN


k-NN with k=5
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

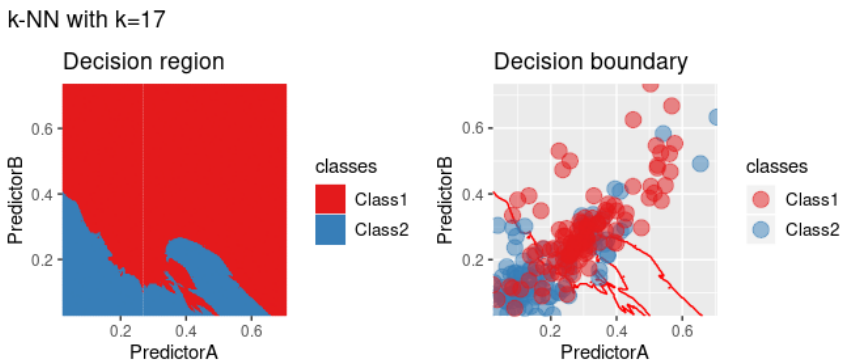
Example: KNN


k-NN with k=9
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

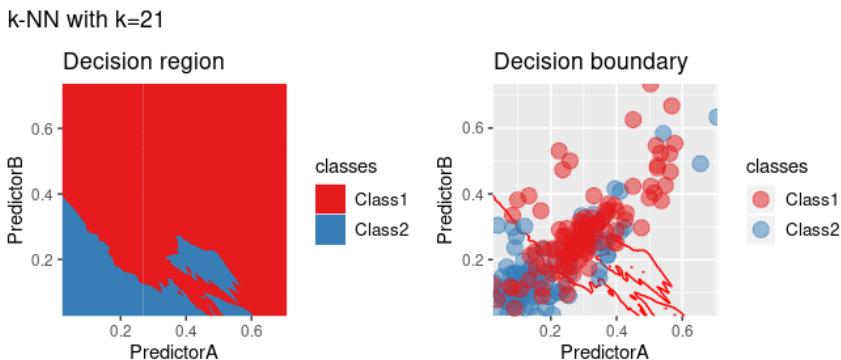
Example: KNN


k-NN with k=13
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

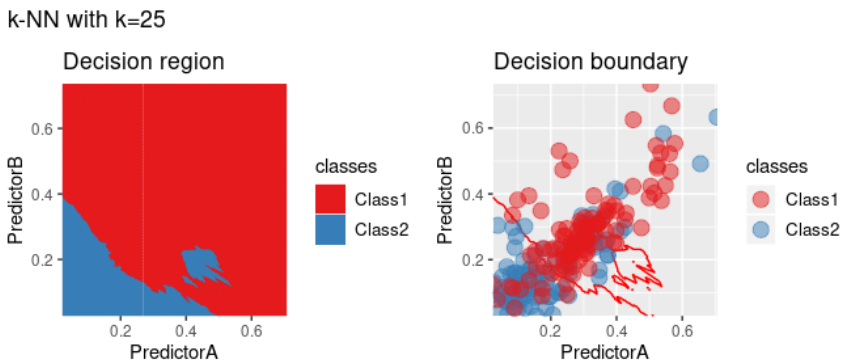
Example: KNN


k-NN with k=17
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

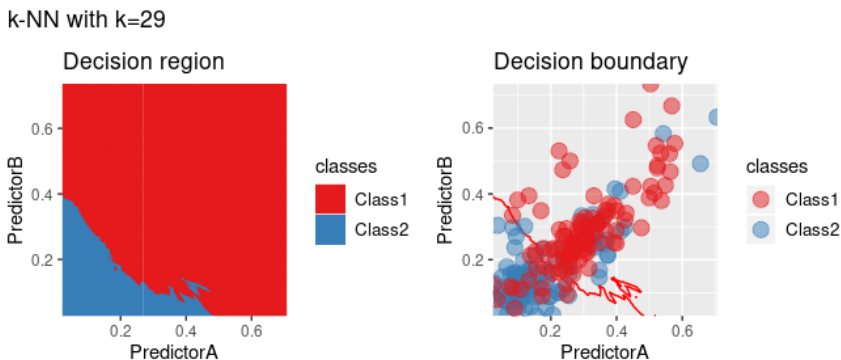
Example: KNN


k-NN with k=21
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

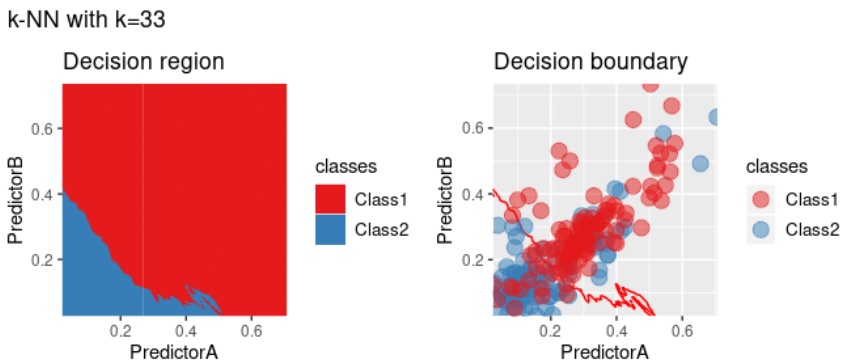
Example: KNN


k-NN with k=25
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

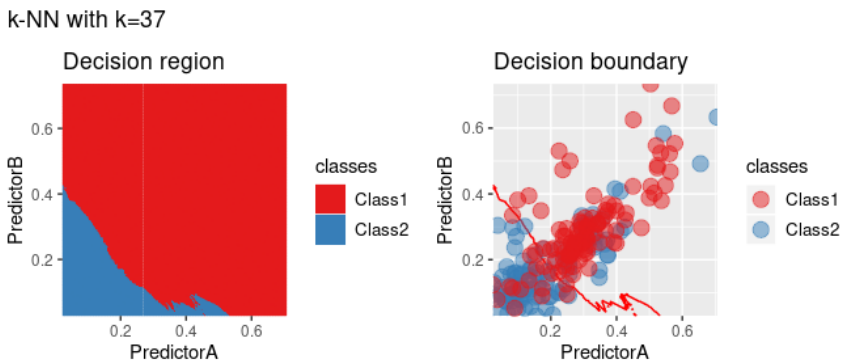
Example: KNN


k-NN with k=29
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

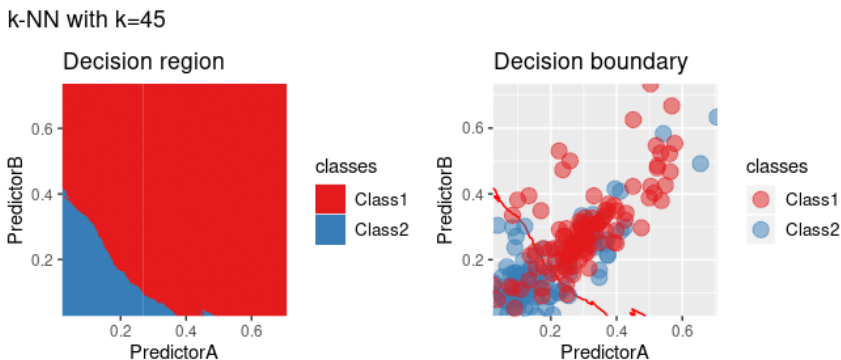
Example: KNN


k-NN with k=33
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

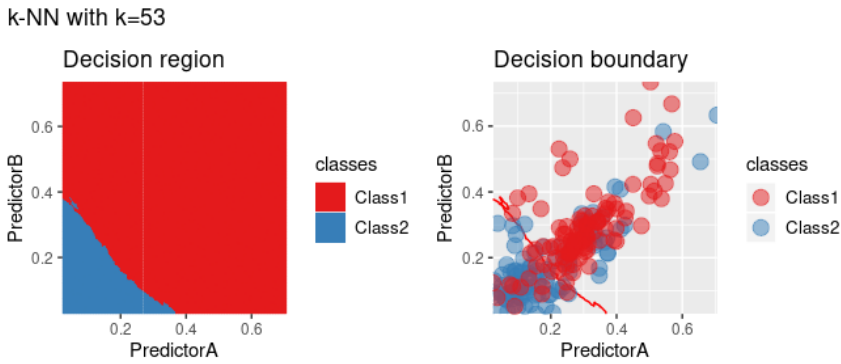
Example: KNN


k-NN with k=37
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

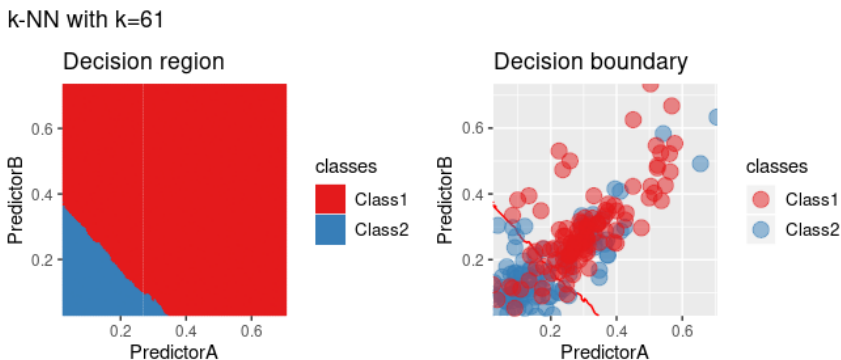
Example: KNN


k-NN with k=45
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

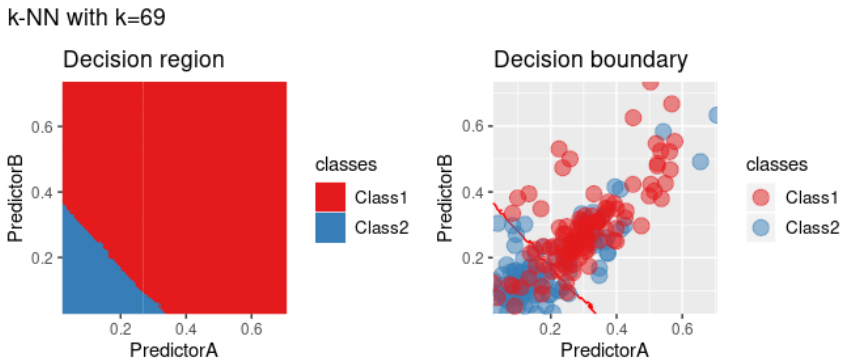
Example: KNN


k-NN with k=53
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

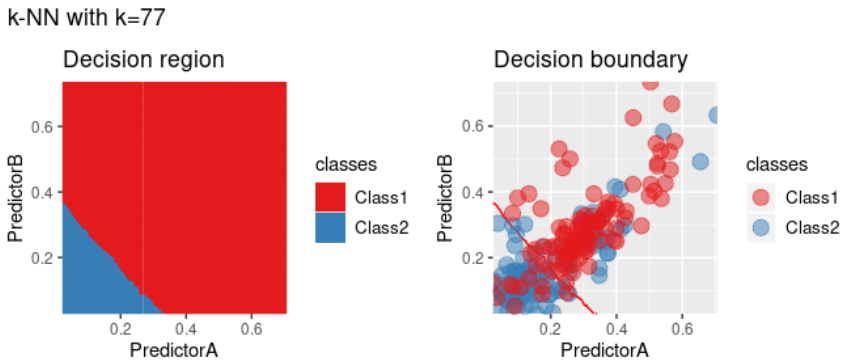
Example: KNN


k-NN with k=61
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

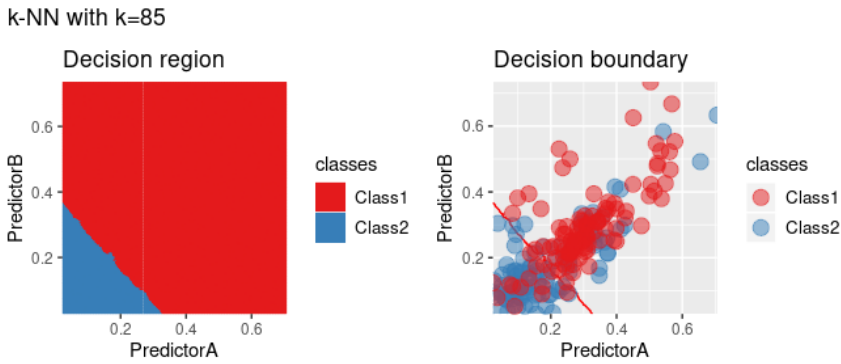
Example: KNN


k-NN with k=69
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

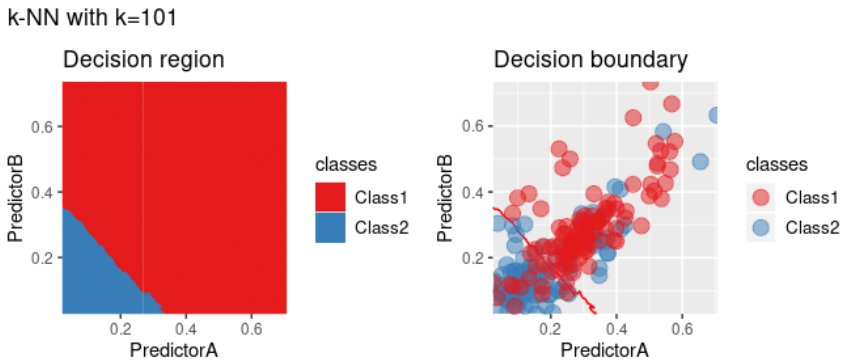
Example: KNN


k-NN with k=77
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

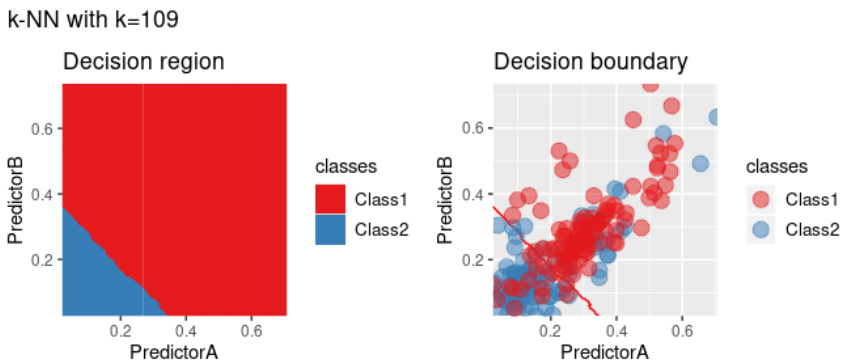
Example: KNN


k-NN with k=85
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

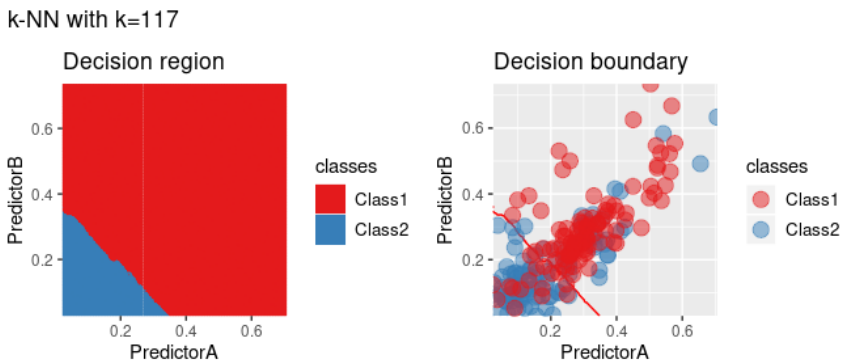
Example: KNN


k-NN with k=101
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

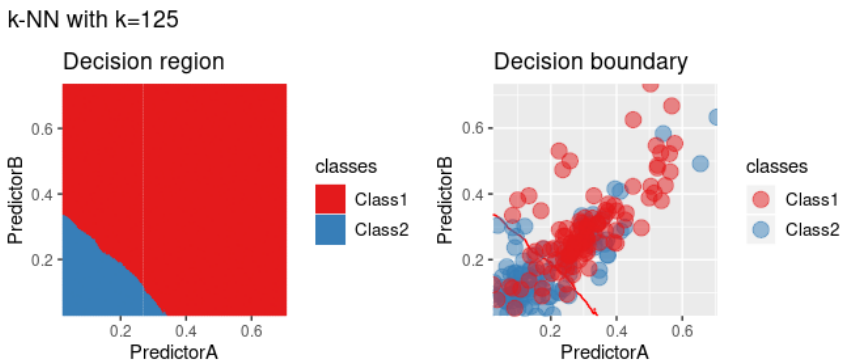
Example: KNN


k-NN with k=109
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

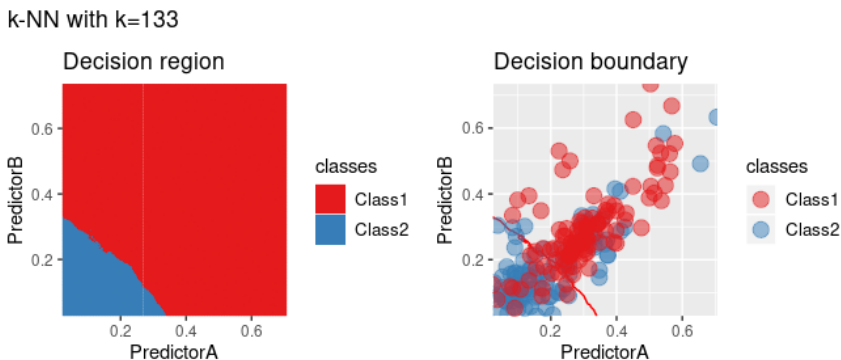
Example: KNN


k-NN with k=117
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

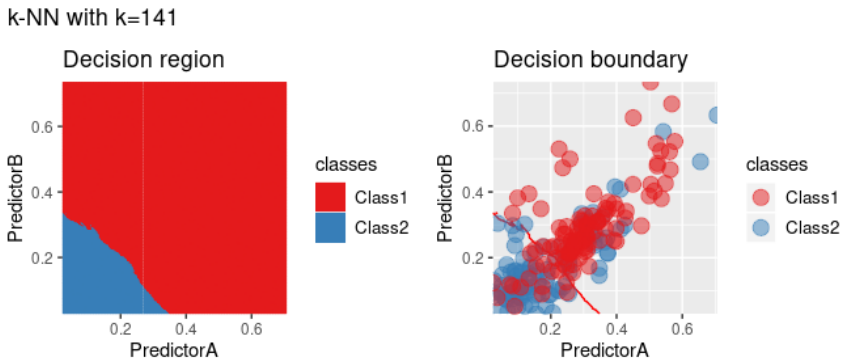
Example: KNN


k-NN with k=125
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

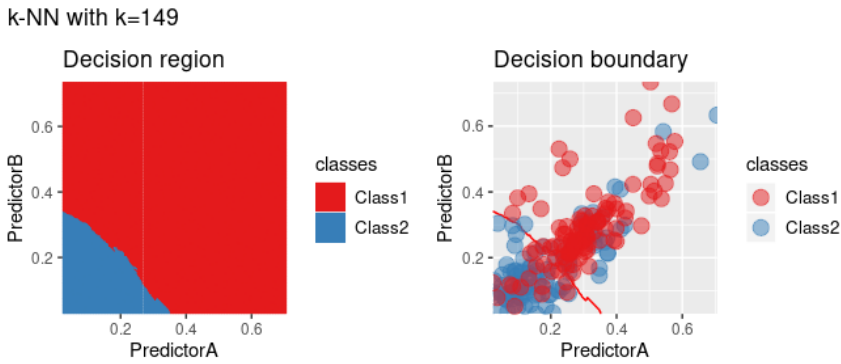
Example: KNN


k-NN with k=133
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

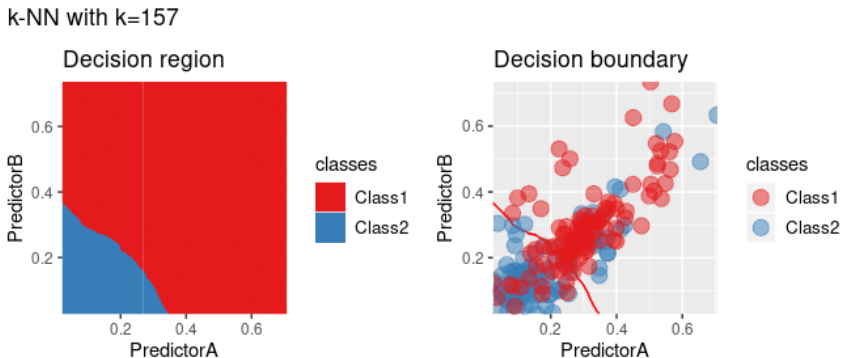
Example: KNN


k-NN with k=141
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

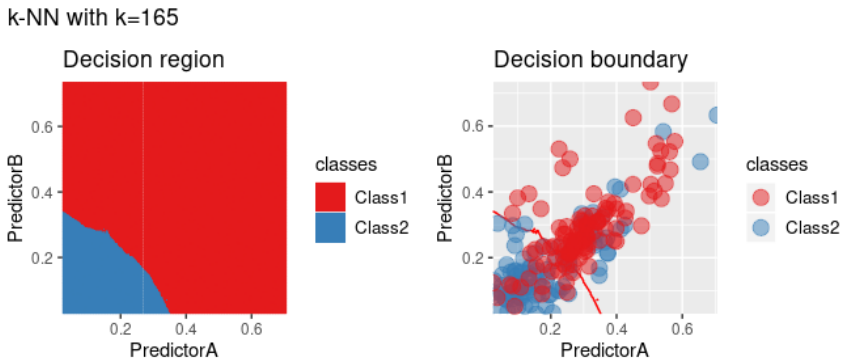
Example: KNN


k-NN with k=149
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

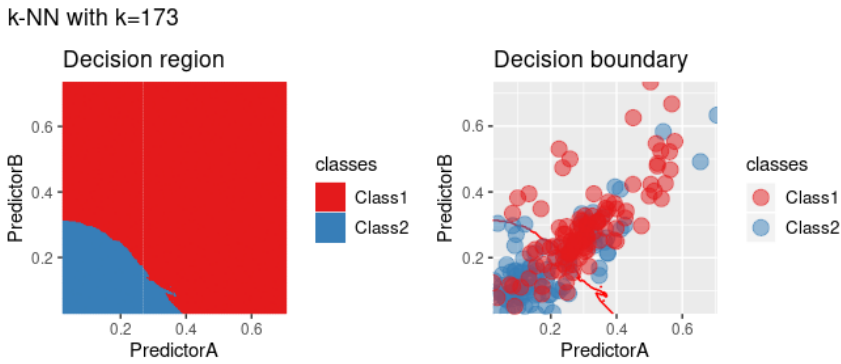
Example: KNN


k-NN with k=157
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

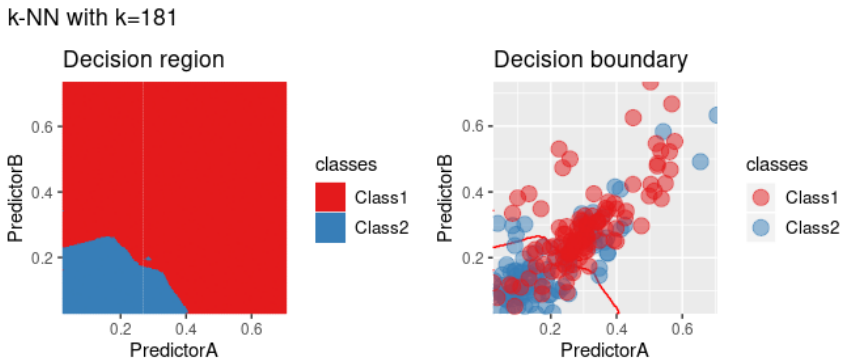
Example: KNN


k-NN with k=165
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

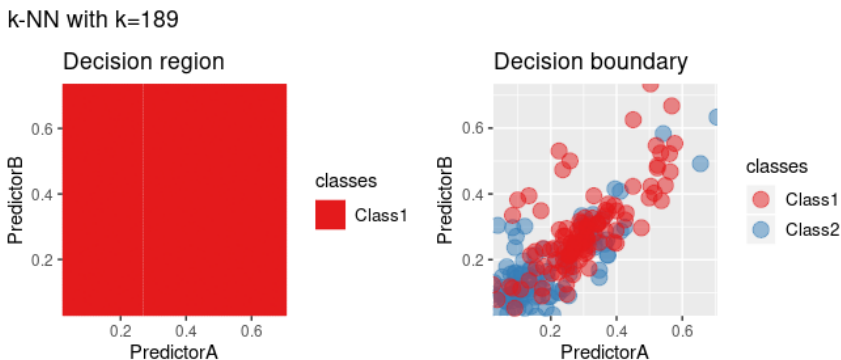
Example: KNN


k-NN with k=173
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

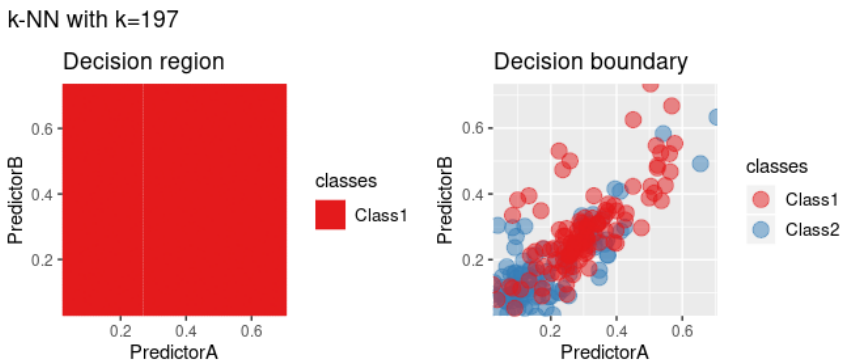
Example: KNN


k-NN with k=181
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

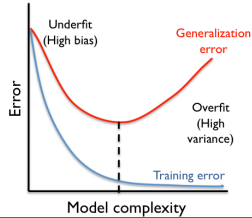
Example: KNN


k-NN with k=189
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: KNN


k-NN with k=197
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Training Risk Issue



Risk behaviour

- Learning/training risk (empirical risk on the learning/training set) decays when the complexity of the **method** increases.
- Quite different behavior when the risk is computed on new observations (generalization risk).

- Overfit for complex methods: parameters learned are too specific to the learning set!
- General situation! (Think of polynomial fit...)
- Need to use a different criterion than the training risk!

Source: JMP

Risk Estimation vs Method Selection

Predictor Risk Estimation

- **Goal:** Given a predictor f assess its quality.
- **Method:** Hold-out risk computation (/ Empirical risk correction).
- **Usage:** Compute an estimate of the risk of a selected f using a **test set** to be used to monitor it in the future.

- Basic block very well understood.

Method Selection

- **Goal:** Given a ML method assess its quality.
- **Method:** Cross Validation (/ Empirical risk correction)
- **Usage:** Compute risk estimates for several ML methods using **training/validation sets** to choose the most promising one.

- Estimates can be pointwise or better intervals.
- Multiple test issues in method selection.

Cross Validation and Empirical Risk Correction

Two Approaches

- **Cross validation:** Use empirical risk criterion but on independent data, very efficient (and almost always used in practice!) but slightly biased as its target uses only a fraction of the data.
- **Correction approach:** use empirical risk criterion but *correct* it with a term increasing with the complexity of $\mathcal{S}$

$$R_n(\widehat{f}_{\mathcal{S}}) \to R_n(\widehat{f}_{\mathcal{S}}) + \text{cor}(\mathcal{S})$$

and choose the method with the smallest corrected risk.

Which loss to use?

- The loss used in the risk: most natural!
- The loss used to estimate $\widehat{\theta}$: penalized estimation!

- Other performance measure can be used.

Cross Validation

Cross Validation

- Use $(1-\epsilon) \times n$ observations to train and $\epsilon \times n$ to verify!
- Possible issues:
  - Validation for a learning set of size $(1-\epsilon) \times n$ instead of n ?
  - Unstable risk estimate if ϵn is too small ?

- Most classical variations:
  - Hold Out,
  - Leave One Out,
  - V-fold cross validation.

Source: M. Kühn

Hold Out

Principle

- Split the dataset $\mathcal{D}$ in 2 sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ of size $n \times (1-\epsilon)$ and $n \times \epsilon$.
- Learn $\widehat{f}^{HO}$ from the subset $\mathcal{D}_{\text{train}}$.
- Compute the empirical risk on the subset $\mathcal{D}_{\text{test}}$:

$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} \ell(Y_i, \widehat{f}^{HO}(\underline{X}_i))$$

Predictor Risk Estimation

- Use $\hat{f}^{HO}$ as predictor.
- Use $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ as an estimate of the risk of this estimator.

Method Selection by Cross Validation

- Compute $\mathcal{R}_n^{HO}(\widehat{f}_{\mathcal{S}}^{HO})$ for all the considered methods,
- Select the method with the smallest CV risk,
- Reestimate the $\widehat{f}_{\mathcal{S}}$ with all the data.

Hold Out

Principle

- Split the dataset $\mathcal{D}$ in 2 sets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ of size $n \times (1-\epsilon)$ and $n \times \epsilon$.
- Learn $\widehat{f}^{HO}$ from the subset $\mathcal{D}_{\text{train}}$.
- Compute the empirical risk on the subset $\mathcal{D}_{\text{test}}$:

$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} \ell(Y_i, \widehat{f}^{HO}(\underline{X}_i))$$

- Only possible setting for risk estimation.

Hold Out Limitation for Method Selection

- Biased toward simpler method as the estimation does not use all the data initially.
- Learning variability of $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ not taken into account.

V-fold Cross Validation

Principle

- Split the dataset $\mathcal{D}$ in V sets $\mathcal{D}_v$ of almost equals size.
- For $v \in \{1, .., V\}$:
  - Learn $\widehat{f}^{-v}$ from the dataset $\mathcal{D}$ minus the set $\mathcal{D}_v$.
  - Compute the empirical risk:

$$\mathcal{R}_n^{-v}(\widehat{f}^{-v}) = \frac{1}{n_v} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_v} \ell(Y_i, \widehat{f}^{-v}(\underline{X}_i))$$

- Compute the average empirical risk:

$$\mathcal{R}_n^{CV}(\widehat{f}) = \frac{1}{V} \sum_{v=1}^{V} \mathcal{R}_n^{-v}(\widehat{f}^{-v})$$

- Estimation of the quality of a method not of a given predictor.
- Leave One Out : V = n.

Source: M. Kühn

V-fold Cross Validation

Analysis (when n is a multiple of V)

- The $\mathcal{R}_n^{-v}(\widehat{f}^{-v})$ are identically distributed variable but are not independent!
- Consequence:

$$\mathbb{E}\left[\mathcal{R}_n^{CV}(\widehat{f})\right] = \mathbb{E}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v})\right]$$

$$\mathbb{V}\text{ar}\left[\mathcal{R}_n^{CV}(\widehat{f})\right] = \frac{1}{V}\mathbb{V}\text{ar}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v})\right] + (1-\frac{1}{V})\,\mathbb{C}\text{ov}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v}), \mathcal{R}_n^{-v'}(\widehat{f}^{-v'})\right]$$

- Average risk for a sample of size $(1-\frac{1}{V})n$.
- Variance term much more complex to analyze!
- Fine analysis shows that the larger V the better...

- Accuracy/Speed tradeoff: $V = 5$ or $V = 10$...

Linear Regression and Leave One Out

- Leave One Out $=$ V fold for $V = n$: very expensive in general.

A fast LOO formula for the linear regression

- **Prop:** for the least squares linear regression,

$$\widehat{f}^{-i}(\underline{X}_i) = \frac{\widehat{f}(\underline{X}_i) - h_{ii} Y_i}{1 - h_{ii}}$$

  with h_{ii} the ith diagonal coefficient of the **hat** (projection) matrix.
- Proof based on linear algebra!
- Leads to a fast formula for LOO:

$$\mathcal{R}_n^{LOO}(\widehat{f}) = \frac{1}{n} \sum_{i=1}^{n} \frac{|Y_i - \widehat{f}(\underline{X}_i)|^2}{(1 - h_{ii})^2}$$

Cross Validation and Confidence Interval

- How to replace pointwise estimation by a confidence interval?
- Can we use the variability of the CV estimates?
- **Negative result:** No unbiased estimate of the variance!

Gaussian Interval (Comparison of the means and $\sim$ indep.)

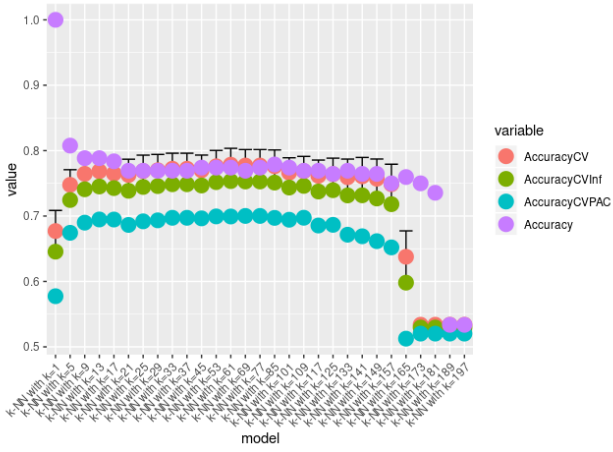
- Compute the empirical variance and divide it by the number of folds to construct an asymptotic Gaussian confidence interval,
- Select the simplest model whose value falls into the confidence interval of the model having the smallest CV risk.

PAC approach (Quantile, $\sim$ indep. and small risk estim. error)

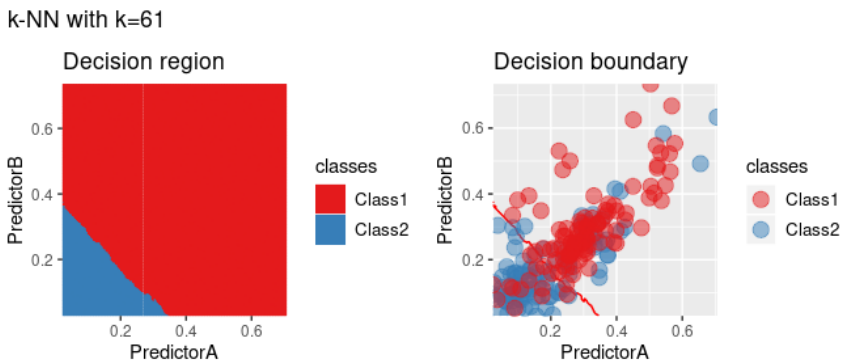
- Compute the raw medians (or a larger raw quantiles)
- Select the model having the smallest quantiles to ensure a small risk with high probability.

- Always reestimate the chosen model with all the data.
- To obtain an unbiased risk estimate of the final predictor: hold out risk on untouched test data.

Cross Validation



Example: KNN ($\widehat{k} = 61$ using cross-validation)



Bootstrap

Risk Estimation and Bootstrap

- Bootstrap train/test splitting:
  - Draw a bootstrap sample $\mathcal{D}_b^{\text{train}}$ of size n (drawn from the original data with replacement) as training set.
  - Use the remaining samples to test $\mathcal{D}_b^{\text{test}} = \mathcal{D} \setminus \mathcal{D}_b^{\text{train}}$.
  - On average $.632n$ distinct samples to train and $.368n$ samples to test.
- Basic bootstrap strategy:
  - Learn $\hat{f}_b$ from $\mathcal{D}_b^{\text{train}}$.
  - Compute a risk estimate on the test:

$$\mathcal{R}_{n,b}(\hat{f}_b) = \frac{1}{|\mathcal{D}_b^{\text{test}}|} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_b^{\text{test}}} \ell(Y_i, \widehat{f}_b(\underline{X}_i))$$

- Looks similar to a 2/3 train and 1/3 test holdout!

Source: S. Raschka

Bootstrap

Repeated Bootstrap Risk Estimation

- Compute several bootstrap risks $\mathcal{R}_{n,b}(\hat{f}_b)$ and average them

$$\mathcal{R}^{Boot}(\hat{f}) = \frac{1}{B}\sum_{b=1}^{B}\mathcal{R}_{n,b}(\hat{f}_b)$$

- Pessimistic (but stable) estimate of the risk as only $.632n$ samples are used to train.

- Bootstrap predictions can be used to assess of the stability!

Source: S. Raschka

Bootstrap

Corrected Bootstrap Risk Estimation

- The training risk is an optimistic risk estimate:

$$\mathcal{R}_n(\hat{f}_b) = \frac{1}{|\mathcal{D}_b^{\text{train}}|} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_b^{\text{train}}} \ell(Y_i, \hat{f}_b(\underline{X}_i))$$

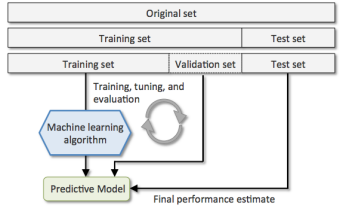
- Combine both estimate for every b:

$$\mathcal{R}'_b(\hat{f}_b) = \omega \mathcal{R}_{n,b}(\hat{f}_b) + (1 - \omega)\mathcal{R}_n(\hat{f}_b)$$

- Choices for ω:
  - .632 rule: set $\omega = .632$
  - .632+ rule: set $\omega = .632/(1 - .368R)$ with $R = (\mathcal{R}_{n,b}(\hat{f}_b) - \mathcal{R}_n(\hat{f}_b))/(\gamma - \mathcal{R}_n(\hat{f}_b))$ where γ is the risk of a predictor trained on the n^2 decoupled data samples $(\underline{X}_i, Y_j)$.
- Works quite well in practice but heuristic justification not obvious.

Source: S. Raschka

Train/Validation/Test



- **Selection Bias Issue:**
  - After method selection, the cross validation is biased.
  - Furthermore, it qualifies the method and not the final predictor.
- Need to (re)estimate the risk of the final predictor.

(Train/Validation)/Test strategy

- **Split** the dataset in two a `(Train/Validation)` and `Test`.
- Use **CV** with the `(Train/Validation)` to **select a method**.
- Train this method on `(Train/Validation)` to **obtain a single predictor**.
- Estimate the **performance of this predictor** on `Test`.

- Every choice made from the data is part of the method!

Source: Shan-Hung Wu & DataLab

Risk Correction

- Empirical loss of an estimator computed on the dataset used to chose it is biased!
- Empirical loss is an optimistic estimate of the true loss.

Risk Correction Heuristic

- Estimate an upper bound of this optimism for a given family.
- Correct the empirical loss by adding this upper bound.

- **Rk:** Finding such an upper bound can be complicated!
- Correction often called a **penalty**.

Penalization

Penalized Loss

- Minimization of

$$\underset{\theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(\underline{X}_i)) + \text{pen}(\theta)$$

where $\text{pen}(\theta)$ is a risk correction (penalty).

Penalties

- Upper bound of the optimism of the empirical loss
- Depends on the loss and the framework!

Instantiation

- Mallows Cp: Least Squares with $\text{pen}(\theta) = 2\frac{d}{n}\sigma^2$.
- AIC Heuristics: Maximum Likelihood with $\text{pen}(\theta) = \frac{d}{n}$.
- BIC Heuristics: Maximum Likelohood with $\text{pen}(\theta) = \log(n)\frac{d}{n}$.
- Structural Risk Minimization: Pred. loss and clever penalty.

Outline

Comparison of Two Means

Means

- **Setting:** r.v. $e_i^{(l)}$ with $1 \leq i \leq n_l$ and $l \in \{1, 2\}$ and their means

$$\overline{e^{(l)}} = \frac{1}{n_l} \sum_{i=1}^{n_l} e_i^{(l)}$$

- **Question:** are the means $\overline{e^{(l)}}$ statistically different?

Classical i.i.d setting

- **Assumption:** $e_i^{(l)}$ are i.i.d. for each l.
- **Test formulation:** Can we reject the null hypothesis that $\mathbb{E}\left[e^{(1)}\right] = \mathbb{E}\left[e^{(2)}\right]$?
- **Methods:**
  - Gaussian (Student) test using asymptotic normality of a mean.
  - Non-parametric permutation test.

- Gaussian approach is linked to confidence intervals.
- The larger n_l the smaller the confidence intervals.

Non i.i.d. case

- **Assumption:** $e_i^{(l)}$ are i.d. for each l but not necessarily independent.
- **Test formulation:** Can we reject the null hypothesis that $\mathbb{E}\left[e^{(1)}\right] = \mathbb{E}\left[e^{(2)}\right]$?
- **Methods:**
  - Gaussian (Student) test using asymptotic normality of a mean but variance is hard to estimate.
  - Non-parametric permutation test but no confidence intervals.

- Setting for Cross Validation (other than holdout).
- Much more complicated than the i.i.d. case

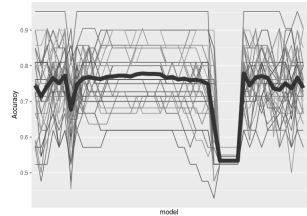
Comparison of Several Means

Several means

- **Assumption:** $e_i^{(l)}$ are i.d. for each l but not necessarily independent.
- **Tests formulation:**
  - Can we reject the null hypothesis that the $\mathbb{E}\left[e^{(l)}\right]$ are different?
  - Is the smaller mean statistically smaller than the second one?
- **Methods:**
  - Gaussian (Student) test using asymptotic normality of a mean with multiple tests correction.
  - Non-parametric permutation test but no confidence intervals.

- Setting for Cross Validation (other than holdout).
- The more models one compares:
  - the larger the confidence intervals
  - the most probable the best model is a lucky winner
- Justify the fallback to the simplest model that could be the best one.

PAC Approach



CV Risk, Methods and Predictors

- Cross-Validation risk: estimate of the average risk of a ML method.
- No risk bound on the predictor obtained in practice.

Probabibly-Approximately-Correct (PAC) Approach

- Replace the control on the average risk by a probabilistic bound

$$\mathbb{P}\Big(\mathbb{E}\Big[\ell(Y, \hat{f}(\underline{X}))\Big] > R\Big) \leq \epsilon$$

- Requires estimating quantiles of the risk.

Cross Validation and Confidence Interval

- How to replace pointwise estimation by a confidence interval?
- Can we use the variability of the CV estimates?
- **Negative result:** No unbiased estimate of the variance!

Gaussian Interval (Comparison of the means and $\sim$ indep.)

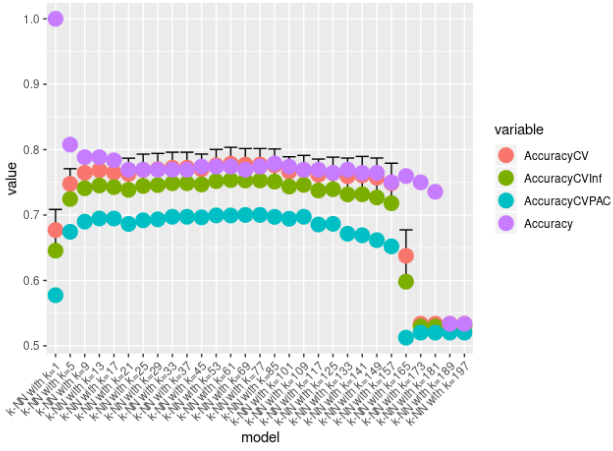
- Compute the empirical variance and divide it by the number of folds to construct an asymptotic Gaussian confidence interval,
- Select the simplest model whose value falls into the confidence interval of the model having the smallest CV risk.

PAC approach (Quantile, $\sim$ indep. and small risk estim. error)

- Compute the raw medians (or a larger raw quantiles)
- Select the model having the smallest quantiles to ensure a small risk with high probability.

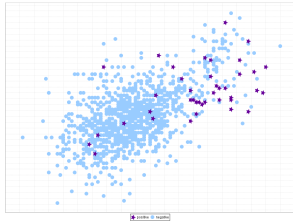
- Always reestimate the chosen model with all the data.
- To obtain an unbiased risk estimate of the final predictor: hold out risk on untouched test data.

Cross Validation



Outline

Unbalanced and Rebalanced Dataset



Unbalanced Class

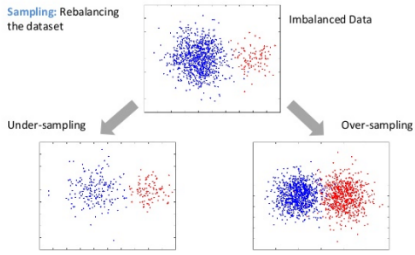
- **Setting:** One of the class is much more present than the other.
- **Issue:** Classifier *too attracted* by the majority class!

Rebalanced Dataset

- **Setting:** Class proportions are different in the training and testing set (stratified sampling)
- **Issue:** Training risks are not estimate of testing risks.

Source: University of Granada

Resampling Strategies



Resampling

- Modify the training dataset so that the classes are more balanced.
- Two flavors:
  - Sub-sampling which spoils data,
  - Over-sampling which needs to create *new* examples.
- **Issues**: Training data is not anymore representative of testing data
- **Hard to do it right!**

Source: Oracle

Resampling Effect

Testing

- Testing class prob.: $\pi_t(k)$
- Testing risk target:
  $\mathbb{E}_{\pi_t}[\ell(Y, f(\underline{X}))] =$
  $\sum_k \pi_t(k)\mathbb{E}[\ell(Y, f(\underline{X}))|Y = k]$

Training

- Training class prob.: $\pi_{tr}(k)$
- Training risk target:
  $\mathbb{E}_{\pi_{tr}}[\ell(Y, f(\underline{X}))] =$
  $\sum_k \pi_{tr}(k)\mathbb{E}[\ell(Y, f(\underline{X}))|Y = k]$

Implicit Testing Risk Using the Training One

- Amounts to use a weighted loss:

$$\begin{aligned}\mathbb{E}_{\pi_{tr}}[\ell(Y, f(\underline{X}))] &= \sum_k \pi_{tr}(k)\mathbb{E}[\ell(Y, f(\underline{X}))|Y = k] \\ &= \sum_k \pi_t(k)\mathbb{E}\left[\frac{\pi_{tr}(k)}{\pi_t(k)}\ell(Y, f(\underline{X}))\middle| Y = k\right] \\ &= \mathbb{E}_{\pi_t}\left[\frac{\pi_{tr}(Y)}{\pi_t(Y)}\ell(Y, f(\underline{X}))\right]\end{aligned}$$

- Put more weight on less probable classes!

Weighted Loss

- In unbalanced situation, often the **cost** of misprediction is not the same for all classes (e.g. medical diagnosis, credit lending...)
- Much better to use this explicitly than to do blind resampling!

Weighted Loss

- **Weighted loss:**

$$\ell(Y, f(\underline{X})) \to C(Y)\ell(Y, f(\underline{X}))$$

- Weighted risk target:

$$\mathbb{E}[C(Y)\ell(Y, f(\underline{X}))]$$

- **Rk:** Strong link with ℓ as C is independent of f.
- Often allow reusing algorithm constructed for ℓ.
- C may also depend on $\underline{X}$...

Weighted Loss, $\ell^{0/1}$ loss and Bayes Classifier

- The Bayes classifier is now:

$$f^\star = \operatorname{argmin} \mathbb{E}[C(Y)\ell(Y, f(\underline{X}))] = \operatorname{argmin} \mathbb{E}_{\underline{X}}\Big[\mathbb{E}_{Y|\underline{X}}[C(Y)\ell(Y, f(\underline{X}))]\Big]$$

Bayes Predictor

- For $\ell^{0/1}$ loss,

$$f^\star(\underline{X}) = \operatorname*{argmax}_k C(k)\mathbb{P}(Y = k|\underline{X})$$

- Same effect than a threshold modification for the binary setting!

- Allow putting more emphasis on some classes than others.

Linking Weights and Proportions

Cost and Proportions

- Testing risk target:
$$\mathbb{E}_{\pi_t}[C_t(Y)\ell(Y, f(\underline{X}))] = \sum_k \pi_t(k)C_t(k)\mathbb{E}[\ell(Y, f(\underline{X}))|Y = k]$$
- Training risk target
$$\mathbb{E}_{\pi_{tr}}[C_{tr}(Y)\ell(Y, f(\underline{X}))] = \sum_k \pi_{tr}(k)C_{tr}(k)\mathbb{E}[\ell(Y, f(\underline{X}))|Y = k]$$
- **Coincide if**
$$\pi_t(k)C_t(k) = \pi_{tr}(k)C_{tr}(k)$$

- Lots of flexibility in the choice of C_t, C_{tr} or π_{tr}.

Combining Weights and Resampling

Weighted Loss and Resampling

- **Weighted loss:** choice of a weight $C_t \neq 1$.
- **Resampling:** use a $\pi_{tr} \neq \pi_t$.

- Stratified sampling may be used to reduce the size of a dataset without loosing a low probability class!

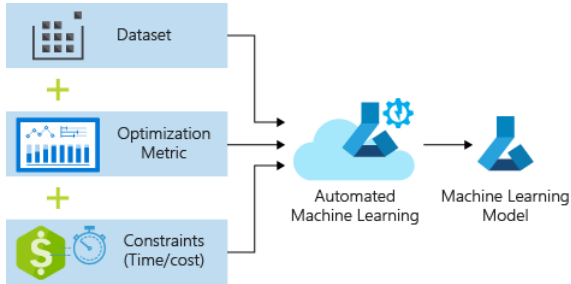
Combining Weights and Resampling

- **Weighted loss:** use $C_{tr} = C_t$ as $\pi_{tr} = \pi_t$.
- **Resampling:** use an implicit $C_t(k) = \pi_{tr}(k)/\pi_t(k)$.
- **Combined:** use $C_{tr}(k) = C_t(k)\pi_t(k)/\pi_{tr}(k)$

- Most ML methods allow such weights!

Outline

Auto ML

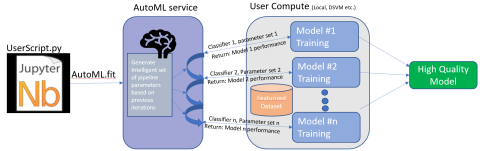


Auto ML

- Automatically propose a good predictor
- Rely heavily on risk evaluations
- **Pros:** easy way to obtain an excellent baseline
- **Cons:** black box that can be abused...

Source: softwareengineeringdaily.com/2019/05/15/introduction-to-automated-machine-learning-automl/

Auto ML Task

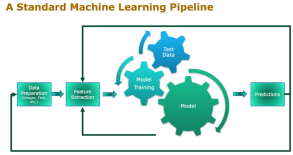


Auto ML Task

- Input:
  - a dataset $\mathcal{D} = (\underline{X}_i, Y_i)$
  - a loss function $\ell(Y, f(\underline{X}))$
  - a set of possible predictors $f_{l,h,\theta}$ corresponding to a method l in a list, with hyperparameters h and parameters θ
- Output:
  - a predictor f equal to $f_{\hat{l},\hat{h},\hat{\theta}}$ or combining several such functions.

Source: Microsoft

Predictors



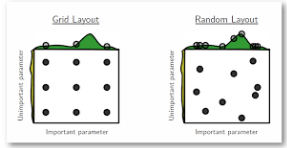
Predictors, a.k.a fitted pipelines

- Preprocessing:
  - Feature design: normalization, coding, kernel...
  - Missing value strategy
  - Feature selection method
- ML Method:
  - Method itself
  - Hyperparameters and architecture
  - Fitted parameters (includes optimization algorithm)

- Quickly amounts to 20 to 50 design decisions!
- **Bruteforce exploration impossible!**

Source: Microsoft

Auto ML and Hyperparameter Optimization

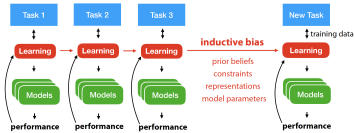


Most Classical Approach of Auto ML

- Task rephrased as an optimization on the discrete/continous space of methods/hyperparameters/parameters.
- Parameters obtained by classical minimization.
- Optimization of methods/hyperparameters much more challenging.
- Approaches:
  - Bruteforce: Grid search and random search
  - Clever exploration: Evolutionary algorithm
  - Surrogate based: Bayesian search and Reinforcement learning

Source: J. Borgstra

Auto ML and Meta-Learning

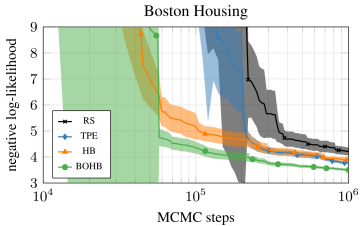


Learn from other Learning Tasks

- Consider the choice of the method from a dataset and a metric as a learning task.
- Requires a way to describe the problems (or to compute a similarity).
- Descriptor often based on a combination of dataset properties and fast method results.
- May output a list of candidates instead of a single method.

- Promising but still quite experimental!

Auto ML and Time Budget



How to obtain a good result with a time constraint?

- Brute force: Time out and methods screening with Meta-Learning (less exploration at the beginning)
- Surrogate based: Bayesian optimization (exploration/exploitation tradeoff)
- Successive elimination: Fast but not accurate performance evaluation at the beginning to eliminate the worst models (more exploration at the beginning)
- Combined strategy: Bandit strategy to obtain a more accurate estimate of risks only for the promising models (exploration/exploitation tradeoff)

Source: A. Biedenkapp

Auto ML benchmark

Benchmark

- Almost always (slightly) better than a good random forest or gradient boosting predictor.
- Worth the try!

Source: AutoML Benchmark

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

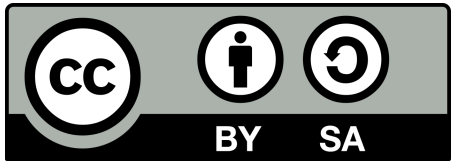
K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

Outline

Supervised Learning

Supervised Learning Framework

- Input measurement $\underline{X} \in \mathcal{X}$
- Output measurement $Y \in \mathcal{Y}$.
- $(\underline{X}, Y) \sim \mathbb{P}$ with $\mathbb{P}$ unknown.
- **Training data** : $\mathcal{D}_n = \{(\underline{X}_1, Y_1), \ldots, (\underline{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbb{P}$)

- Often
  - $\underline{X} \in \mathbb{R}^d$ and $Y \in \{-1, 1\}$ (classification)
  - or $\underline{X} \in \mathbb{R}^d$ and $Y \in \mathbb{R}$ (regression).
- A **predictor** is a function in $\mathcal{F} = \{f : \mathcal{X} \to \mathcal{Y} \text{ meas.}\}$

Goal

- Construct a **good** predictor $\widehat{f}$ from the training data.

- Need to specify the meaning of good.
- Classification and regression are almost the **same** problem!

Loss and Probabilistic Framework

Loss function for a generic predictor

- **Loss function**: $\ell(Y, f(\underline{X}))$ measures the goodness of the prediction of Y by $f(\underline{X})$
- Examples:
  - 0/1 loss: $\ell(Y, f(\underline{X})) = \mathbf{1}_{Y \neq f(\underline{X})}$
  - Quadratic loss: $\ell(Y, f(\underline{X})) = |Y - f(\underline{X})|^2$

Risk function

- Risk measured as the average loss for a new couple:

$$\mathcal{R}(f) = \mathbb{E}_{(\underline{X},Y)\sim\mathbb{P}}[\ell(Y, f(\underline{X}))]$$

- Examples:
  - 0/1 loss: $\mathbb{E}[\ell(Y, f(\underline{X}))] = \mathbb{P}(Y \neq f(\underline{X}))$
  - Quadratic loss: $\mathbb{E}[\ell(Y, f(\underline{X}))] = \mathbb{E}\left[|Y - f(\underline{X})|^2\right]$

- **Beware:** As $\widehat{f}$ depends on $\mathcal{D}_n$, $\mathcal{R}(\widehat{f})$ is a random variable!

Goal

Machine Learning

- Learn a rule to construct a **predictor** $\widehat{f} \in \mathcal{F}$ from the training data $\mathcal{D}_n$ s.t. **the risk** $\mathcal{R}(\widehat{f})$ is **small on average** or with high probability with respect to $\mathcal{D}_n$.

- In practice, the rule should be an algorithm!

Canonical example: Empirical Risk Minimizer

- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the empirical loss

$$\widehat{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f_\theta(\underline{X}_i))$$

- Examples:
  - Linear regression
  - Linear classification with

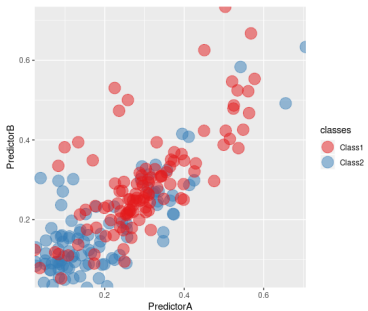
$$\mathcal{S} = \{\underline{x} \mapsto \operatorname{sign}\{\underline{x}^\top \beta + \beta^{(0)}\} \, / \beta \in \mathbb{R}^d, \beta^{(0)} \in \mathbb{R}\}$$

Example: TwoClass Dataset

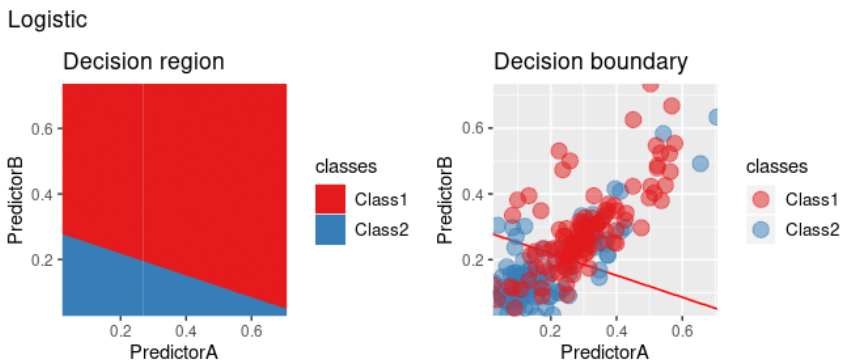
Synthetic Dataset

- Two features/covariates.
- Two classes.

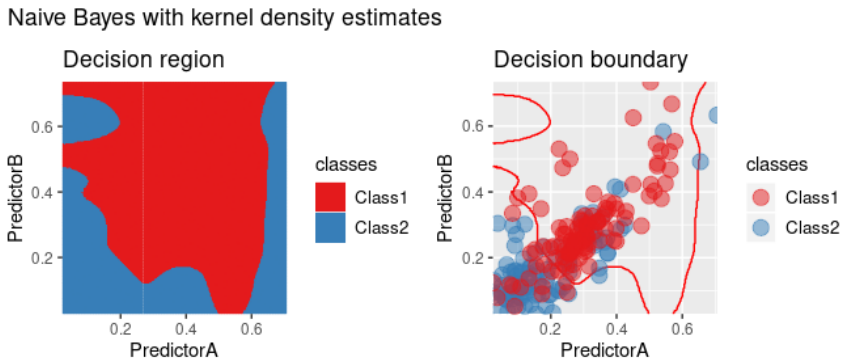
- Dataset from *Applied Predictive Modeling*, M. Kuhn and K. Johnson, Springer
- Numerical experiments with `R` and the {`caret`} package.



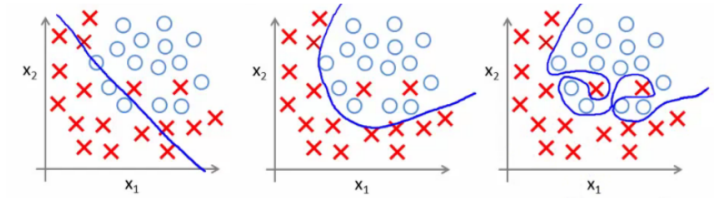
Example: Linear Discrimination



Example: More Complex Model



Under-fitting / Over-fitting Issue

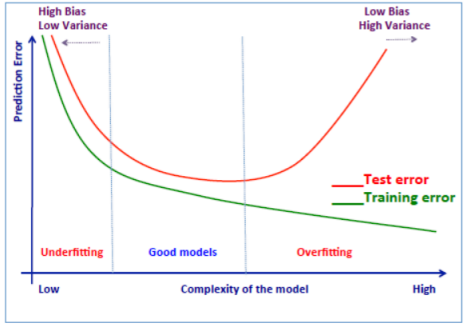


Model Complexity Dilemna

- What is best a simple or a complex model?
- Too simple to be good? Too complex to be learned?

Source: A. Ng

Under-fitting / Over-fitting Issue

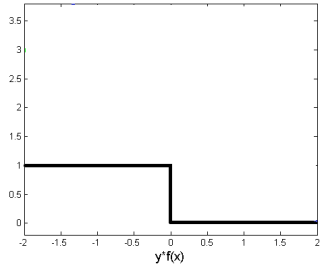


Under-fitting / Over-fitting

- **Under-fitting:** simple model are too simple.
- **Over-fitting:** complex model are too specific to the training set.

Source: Unknown

Binary Classification Loss Issue

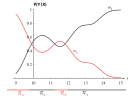


Empirical Risk Minimizer

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

- Classification loss: $\ell^{0/1}(y, f(\underline{x})) = \mathbf{1}_{y \neq f(\underline{x})}$
- Not convex and not smooth!

Probabilistic Point of View
Ideal Solution and Estimation



- The best solution $f^\star$ (which is independent of $\mathcal{D}_n$) is

$$f^\star = \arg\min_{f \in \mathcal{F}} \mathcal{R}(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}[\ell(Y, f(\underline{X}))] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\Big[\mathbb{E}_{Y|\underline{X}}[\ell(Y, f(\underline{x}))]\Big]$$

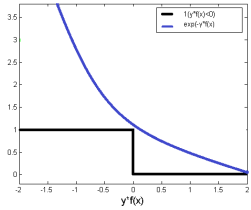
Bayes Predictor (explicit solution)

In binary classification with $0-1$ loss:

$$f^\star(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- **Issue:** Solution requires to **know** $\mathbb{E}[Y|\underline{X}]$ for all values of $\underline{X}$!
- **Solution:** Replace it by an estimate.

Optimization Point of View
Loss Convexification



Minimizer of the risk

$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

- **Issue:** Classification loss is not convex or smooth.
- **Solution:** Replace it by a convex majorant.

Probabilistic and Optimization Framework

How to find a good function f with a *small* risk

$$\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\underline{X}))] \quad ?$$

Canonical approach: $\widehat{f}_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\underline{X}_i))$

Problems

- How to choose $\mathcal{S}$?
- How to compute the minimization?

A Probabilistic Point of View

Solution: For $\underline{X}$, estimate $Y|\underline{X}$ plug this estimate in the Bayes classifier:
(Generalized) Linear Models, Kernel methods, k-nn, Naive Bayes, Tree, Bagging...

An Optimization Point of View

Solution: If necessary replace the loss ℓ by an upper bound $\bar{\ell}$ and minimize the empirical loss: **SVR, SVM, Neural Network,Tree, Boosting...**

Outline

Three Classical Methods in a Nutshell

Logistic Regression

- Let $f_\theta(\underline{X}) = \underline{X}^\top \beta + \beta^{(0)}$ with $\theta = (\beta, \beta^{(0)})$.
- Let $\mathbb{P}_\theta(Y = 1|\underline{X}) = e^{-f_\theta(\underline{X})}/(1 + e^{f_\theta(\underline{X})})$
- Estimate θ by $\hat{\theta}$ using a Maximum Likelihood.
- Classify using $\mathbb{P}_{\hat{\theta}}(Y = 1|\underline{X}) > 1/2$

k Nearest Neighbors

- For any $\underline{X}'$, define $\mathcal{V}_{\underline{X}'}$ as the k closest samples X_i from the dataset.
- Compute a score $g_k = \sum_{X_i \in \mathcal{V}_{\underline{X}'}} \mathbf{1}_{Y_i = k}$
- Classify using $\arg\max g_k$ (majority vote).

Three Classical Methods in a Nutshell

Quadratic Discrimant Analysis

- For each class, estimate the mean μ_k and the covariance matrix Σ_k.
- Estimate the proportion $\mathbb{P}(Y = k)$ of each class.
- Compute a score $\ln(\mathbb{P}(\underline{X}|Y = k)) + \ln(\mathbb{P}(Y = k))$

$$g_k(\underline{X}) = -\frac{1}{2}(\underline{X} - \mu_k)^\top \Sigma_k^{-1}(\underline{X} - \mu_k) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\mathbb{P}(Y = k))$$

- Classify using $\arg\max g_k$

- Those three methods rely on a similar heuristic: the probabilistic point of view!

Best Solution

- The best solution $f^\star$ (which is independent of $\mathcal{D}_n$) is

$$f^\star = \arg\min_{f \in \mathcal{F}} R(f) = \arg\min_{f \in \mathcal{F}} \mathbb{E}[\ell(Y, f(\underline{X}))] = \arg\min_{f \in \mathcal{F}} \mathbb{E}_{\underline{X}}\Big[\mathbb{E}_{Y|\underline{X}}[\ell(Y, f(\underline{X})]\Big]$$

Bayes Predictor (explicit solution)

- In binary classification with $0-1$ loss:

$$f^\star(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y=+1|\underline{X}) \geq \mathbb{P}(Y=-1|\underline{X}) \\ & \Leftrightarrow \mathbb{P}(Y=+1|\underline{X}) \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

Issue: Explicit solution requires to **know** $Y|\underline{X}$ for all values of $\underline{X}$!

Plugin Predictor

- **Idea:** Estimate $Y|\underline{X}$ by $\widehat{Y|\underline{X}}$ and plug it the Bayes classifier.

Plugin Bayes Predictor

- In binary classification with $0-1$ loss:

$$\widehat{f}(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{\mathbb{P}(Y=+1|\underline{X})} \geq \widehat{\mathbb{P}(Y=-1|\underline{X})} \\ & \Leftrightarrow \widehat{\mathbb{P}(Y=+1|\underline{X})} \geq 1/2 \\ -1 & \text{otherwise} \end{cases}$$

Plugin Predictor

Three main heuristics

- **Parametric Conditional modeling:** Estimate the law of $Y|\underline{X}$ by a **parametric** law $\mathcal{L}_\theta(\underline{X})$: *Logistic regression*
- **Non Parametric Conditional modeling:** Estimate the law of $Y|\underline{X}$ by a **non parametric** estimate: *kernel methods, nearest neighbors...*
- **Fully Generative modeling:** Estimate the law of $(\underline{X}, Y)$ and use the **Bayes formula** to deduce an estimate of $Y|\underline{X}$: *LDA/QDA, Naive Bayes, Gaussian Processes*

Plugin Classifier

- **Input**: a data set $\mathcal{D}_n$
  Learn $Y|\underline{X}$ or equivalently $\mathbb{P}(Y = k|\underline{X})$ (using the data set) and plug this estimate in the Bayes classifier
- **Output**: a classifier $\widehat{f} : \mathbb{R}^d \to \{-1, 1\}$

$$\widehat{f}(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(\widehat{Y = 1}|\underline{X}) \geq \mathbb{P}(\widehat{Y = -1}|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- Can we guaranty that the classifier is good if $Y|\underline{X}$ is well estimated?

Classification Risk Analysis

Theorem

- If $\widehat{f} = \text{sign}(2\widehat{p}_{+1} - 1)$ then

$$\begin{aligned}\mathbb{E}\left[\ell^{0,1}(Y, \widehat{f}(\underline{X}))\right] - \mathbb{E}\left[\ell^{0,1}(Y, f^\star(\underline{X}))\right] &\leq \mathbb{E}\left[\|\widehat{Y|\underline{X}} - Y|\underline{X}\|_1\right] \\ &\leq \left(\mathbb{E}\left[2\text{KL}(Y|\underline{X}, \widehat{Y|\underline{X}}\right]\right)^{1/2}\end{aligned}$$

- If one estimates $\mathbb{P}(Y = 1|\underline{X})$ well then one estimates $f^\star$ well!
- Link between a *conditional density estimation* task and a *classification* one!
- **Rk:** In general, the conditional density estimation task is more complicated as one should be good for all values of $\mathbb{P}(Y = 1|\underline{X})$ while the classification task focus on values around $1/2$ for the $0/1$ loss!
- In **regression**, (often) direct control of the quadratic loss...

Outline

Logistic Modeling

- Direct modeling of $Y|\underline{x}$.

The Binary logistic model ($Y \in \{-1, 1\}$)

$$\mathbb{P}(Y = 1|\underline{x}) = \frac{e^{\phi(\underline{x})^\top \beta}}{1 + e^{\phi(\underline{x})^\top \beta}}$$

where $\phi(\underline{x})$ is a transformation of the individual $\underline{x}$

- In this model, one verifies that $\mathbb{P}(Y = 1|\underline{x}) \geq \mathbb{P}(Y = -1|\underline{x}) \quad \Leftrightarrow \quad \phi(\underline{x})^\top \beta \geq 0$
- True $Y|x$ may not belong to this model $\Rightarrow$ maximum likelihood of β only finds a good approximation!
- Binary Logistic classifier:

$$\widehat{f}_L(\underline{x}) = \begin{cases} +1 & \text{if } \phi(\underline{x})^\top \widehat{\beta} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

where $\widehat{\beta}$ is estimated by maximum likelihood.

Logistic Modeling

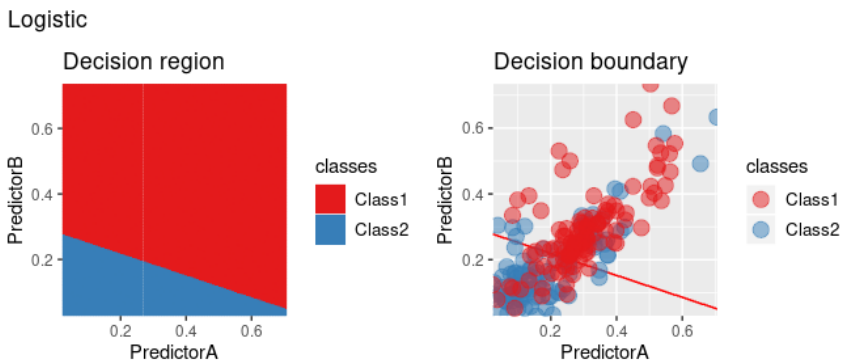
- Logistic model: approximation of $\mathcal{B}(\mathbb{P}(Y=1|\underline{x}))$ by $\mathcal{B}(h(\phi(\underline{x})^\top\beta))$ with $h(t)=\frac{e^t}{1+e^t}$.

Negative log-likelihood formula

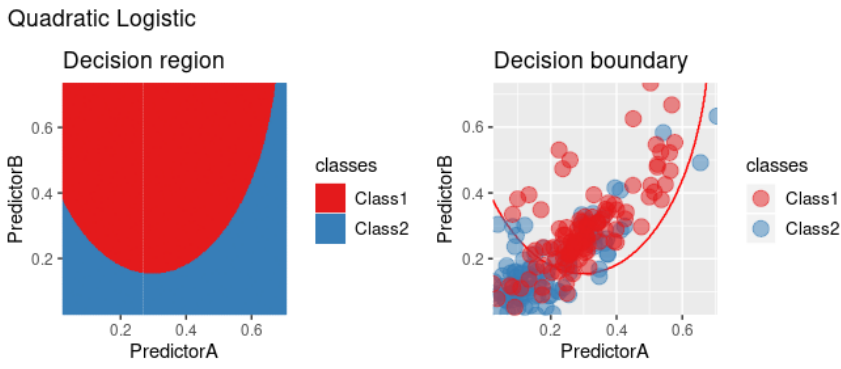
$$
\begin{aligned}
&-\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{y_i=1}\log(h(\phi(\underline{x}_i)^\top\beta))+\mathbf{1}_{y_i=-1}\log(1-h(\phi(\underline{x}_i)^\top\beta))\right)\\
&=-\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{1}_{y_i=1}\log\frac{e^{\phi(\underline{x}_i)^\top\beta}}{1+e^{\phi(\underline{x}_i)^\top\beta}}+\mathbf{1}_{y_i=-1}\log\frac{1}{1+e^{\phi(\underline{x}_i)^\top\beta}}\right)\\
&=\frac{1}{n}\sum_{i=1}^{n}\log\left(1+e^{-y_i(\phi(\underline{x}_i)^\top\beta)}\right)
\end{aligned}
$$

- Convex function in β.
- **Remark:** You can also use your favorite parametric model instead of the logistic one...

Example: Logistic

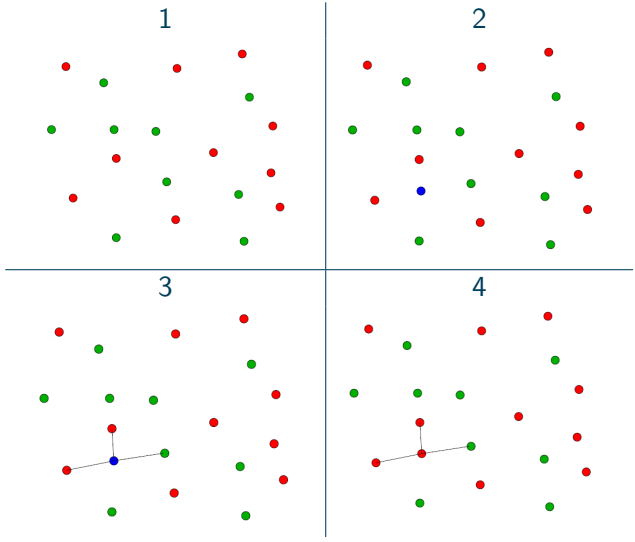


Example: Quadratic Logistic



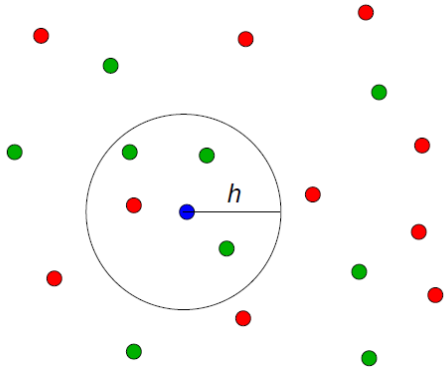
Outline

Example: k Nearest-Neighbors (with $k = 3$)



Source: Unknown

Example: k Nearest-Neighbors (with $k = 4$)



Source: Unknown

k Nearest-Neighbors

- Neighborhood $\mathcal{V}_{\underline{x}}$ of $\underline{x}$: k learning samples closest from $\underline{x}$.

k-NN as local conditional density estimate

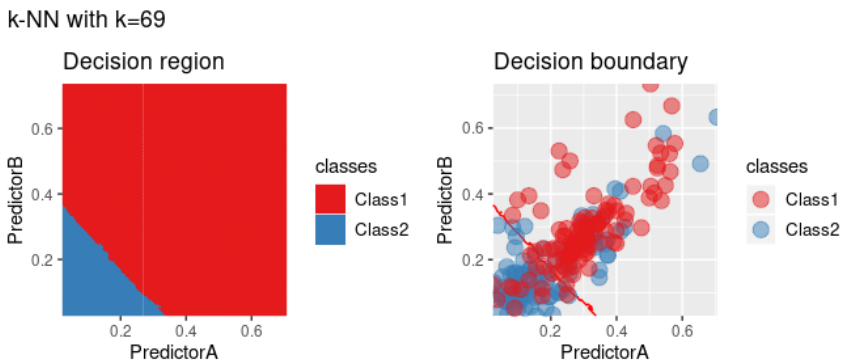
$$\widehat{\mathbb{P}(Y=1|\underline{X})} = \frac{\sum_{\underline{X}_i \in \mathcal{V}_{\underline{X}}} \mathbf{1}_{\{Y_i=+1\}}}{|\mathcal{V}_{\underline{X}}|}$$

- KNN Classifier:

$$\widehat{f}_{KNN}(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{\mathbb{P}(Y=1|\underline{X})} \geq \widehat{\mathbb{P}(Y=-1|\underline{X})} \\ -1 & \text{otherwise} \end{cases}$$

- **Lazy learning**: all the computations have to be done at prediction time.
- **Remark:** You can also use your favorite kernel estimator...

Example: KNN


k-NN with k=69
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Outline

Fully Generative Modeling

- **Idea:** If one knows the law of $(\underline{X}, Y)$ everything is easy!

Bayes formula

- With a slight abuse of notation,

$$\begin{aligned}\mathbb{P}(Y|\underline{X}) &= \frac{\mathbb{P}((\underline{X}, Y))}{\mathbb{P}(\underline{X})} \\ &= \frac{\mathbb{P}(\underline{X}|Y)\,\mathbb{P}(Y)}{\mathbb{P}(\underline{X})}\end{aligned}$$

- **Generative Modeling:**
  - Propose a model for $(\underline{X}, Y)$ (or equivalently $\underline{X}|Y$ and Y),
  - Estimate it as a density estimation problem,
  - Plug the estimate in the Bayes formula
  - Plug the conditional estimate in the Bayes *classifier*.
- **Rk:** Require to estimate $(\underline{X}, Y)$ rather than only $Y|\underline{X}$!
- Great flexibility in the model design but may lead to complex computation.

Fully Generative Modeling

- Simpler setting in classification!

Bayes formula

$$\mathbb{P}(Y = k|\underline{X}) = \frac{\mathbb{P}(\underline{X}|Y = k)\,\mathbb{P}(Y = k)}{\mathbb{P}(\underline{X})}$$

- Binary Bayes classifier (the best solution)

$$f^\star(\underline{X}) = \begin{cases} +1 & \text{if } \mathbb{P}(Y = 1|\underline{X}) \geq \mathbb{P}(Y = -1|\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- **Heuristic**: Estimate those quantities and plug the estimations.
- By using different models/estimators for $\mathbb{P}(\underline{X}|Y)$, we get different classifiers.
- **Rk:** No need to renormalize by $\mathbb{P}(\underline{X})$ to take the decision!

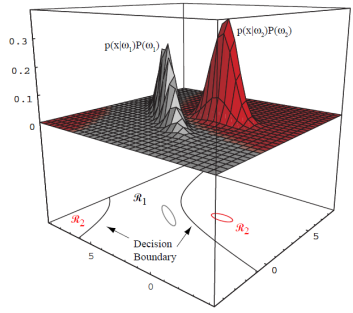
Discriminant Analysis

Discriminant Analysis (Gaussian model)

- The densities are modeled as multivariate normal, i.e.,
$$\mathbb{P}(\underline{X}|Y=k) \sim \mathcal{N}_{\mu_k,\Sigma_k}$$
- Discriminant functions: $\boldsymbol{g_k(\underline{X}) = \ln(\mathbb{P}(\underline{X}|Y=k)) + \ln(\mathbb{P}(Y=k))}$
$$g_k(\underline{X}) = -\frac{1}{2}(\underline{X}-\mu_k)^\top \Sigma_k^{-1}(\underline{X}-\mu_k) - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln(|\Sigma_k|) + \ln(\mathbb{P}(Y=k))$$
- QDA (different Σ_k in each class) and LDA ($\Sigma_k = \Sigma$ for all k)

- **Beware: this model can be false but the methodology remains valid!**

Discriminant Analysis

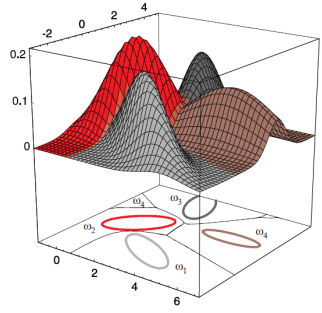


Quadratic Discriminant Analysis

- The probability densities are Gaussian
- The effect of any decision rule is to divide the feature space into some decision regions $\mathcal{R}_1, \mathcal{R}_2$
- The regions are separated by decision boundaries

Source: A. Fermin

Discriminant Analysis



Quadratic Discriminant Analysis

- The probability densities are Gaussian
- The effect of any decision rule is to divide the feature space into some decision regions $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_c$
- The regions are separated by decision boundaries

Source: A. Fermin

Estimation

In practice, we will need to estimate μ_k, Σ_k and $\mathbb{P}_k := \mathbb{P}(Y = k)$

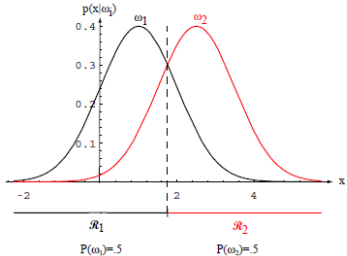
- The estimate proportion $\widehat{\mathbb{P}(Y = k)} = \frac{n_k}{n} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{\{Y_i = k\}}$
- Maximum likelihood estimate of $\widehat{\mu_k}$ and $\widehat{\Sigma_k}$ (explicit formulas)

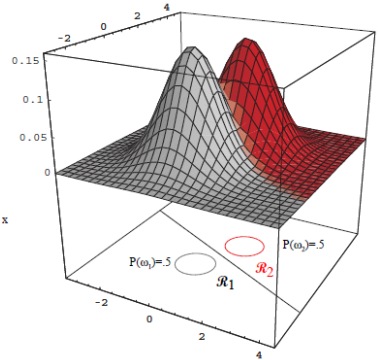
- DA classifier

$$\widehat{f}_G(\underline{X}) = \begin{cases} +1 & \text{if } \widehat{g}_{+1}(\underline{X}) \geq \widehat{g}_{-1}(\underline{X}) \\ -1 & \text{otherwise} \end{cases}$$

- Decision boundaries: quadratic = degree 2 polynomials.
- If one imposes $\Sigma_{-1} = \Sigma_1 = \Sigma$ then the decision boundaries is a linear hyperplane.

Discriminant Analysis



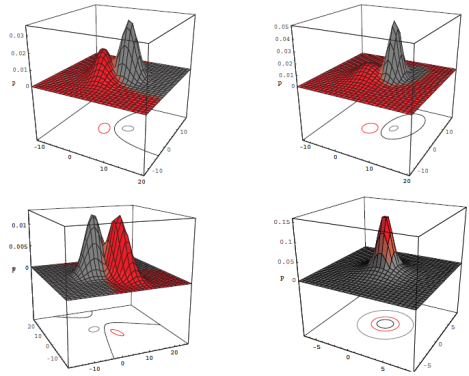


Linear Discriminant Analysis

- $\Sigma_{\omega_1} = \Sigma_{\omega_2} = \Sigma$
- The decision boundaries are linear hyperplanes

Source: A. Fermin

Discriminant Analysis

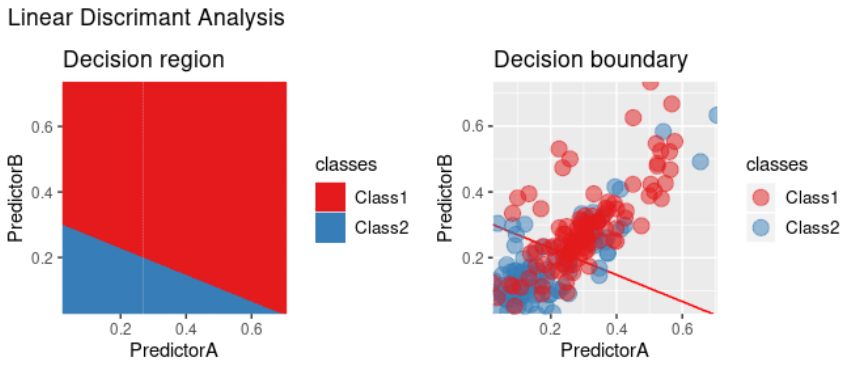


Quadratic Discriminant Analysis

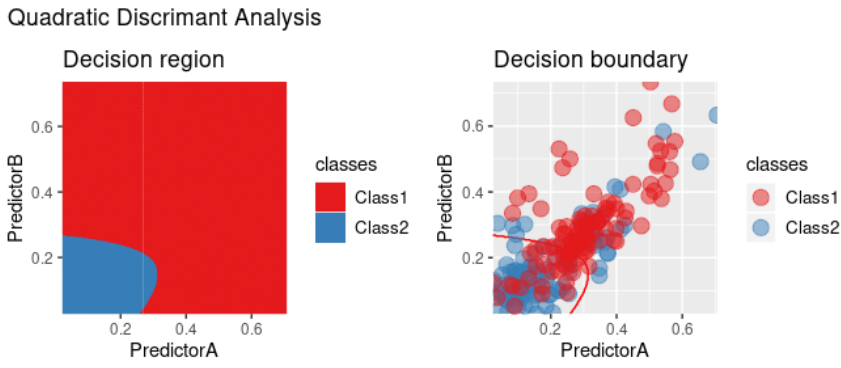
- $\Sigma_{\omega_1} \neq \Sigma_{\omega_2}$
- Arbitrary Gaussian distributions lead to Bayes decision boundaries that are general quadratics.

Source: A. Fermin

Example: LDA


Linear Discrimant Analysis
Decision region
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2
Decision boundary
PredictorB
0.6
0.4
0.2
0.2
0.4
0.6
PredictorA
classes
Class1
Class2

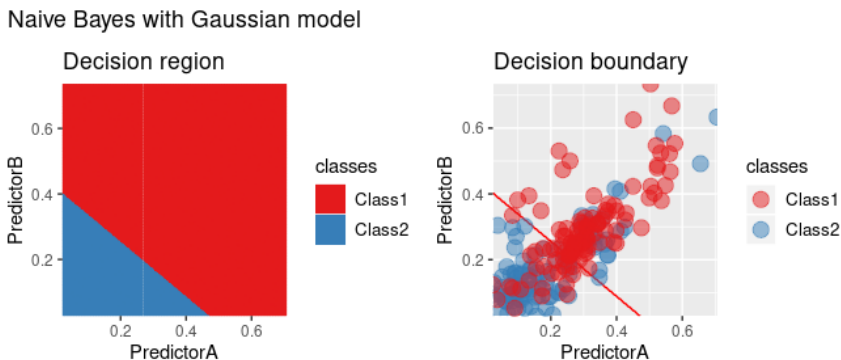
Example: QDA


Quadratic Discrimant Analysis
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

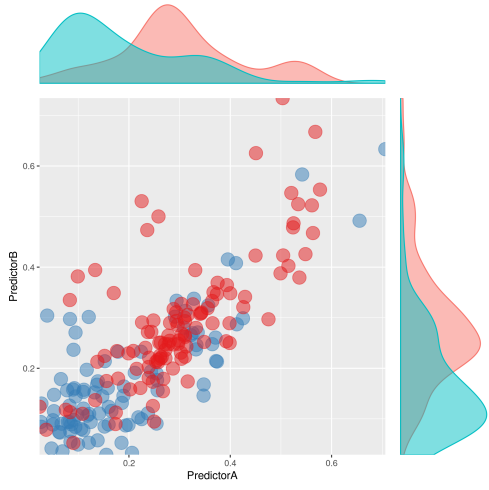
Naive Bayes

- Classical algorithm using a crude modeling for $\mathbb{P}(\underline{X}|Y)$:
  - Feature **independence** assumption:
$$\mathbb{P}(\underline{X}|Y) = \prod_{l=1}^{d} \mathbb{P}\left(\underline{X}^{(l)}\middle|Y\right)$$
  - Simple featurewise model: binomial if binary, multinomial if finite and Gaussian if continuous
- If all features are continuous, similar to the previous Gaussian but with a **diagonal covariance matrix**!
- Very simple learning even in **very high dimension!**

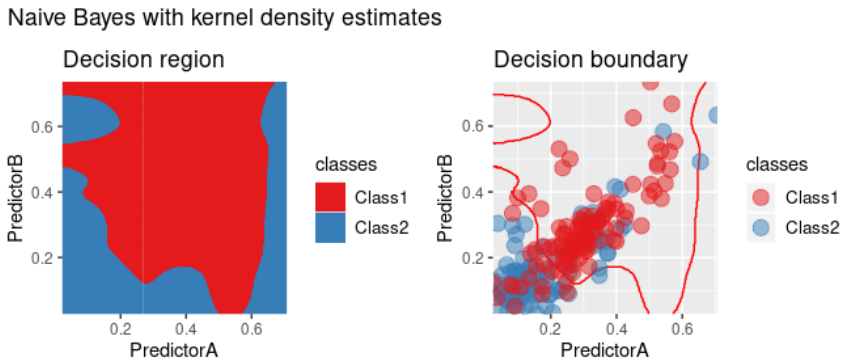
Example: Naive Bayes



Naive Bayes with Density Estimation



Example: Naive Bayes



Outline

Probabilistic and Optimization Framework

How to find a good function f with a *small* risk

$$\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\underline{X}))] \quad ?$$

Canonical approach: $\widehat{f}_{\mathcal{S}} = \text{argmin}_{f \in \mathcal{S}} \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\underline{X}_i))$

Problems

- How to choose $\mathcal{S}$?
- How to compute the minimization?

A Probabilistic Point of View

Solution: For $\underline{X}$, estimate $Y|\underline{X}$ plug this estimate in the Bayes classifier:
(Generalized) Linear Models, Kernel methods, k-nn, Naive Bayes, Tree, Bagging...

An Optimization Point of View

Solution: If necessary replace the loss ℓ by an upper bound $\bar{\ell}$ and minimize the empirical loss: **SVR, SVM, Neural Network,Tree, Boosting...**

Empirical Risk Minimization

- The best solution $f^\star$ is the one minimizing

$$f^\star = \arg\min R(f) = \arg\min \mathbb{E}[\ell(Y, f(\underline{X}))]$$

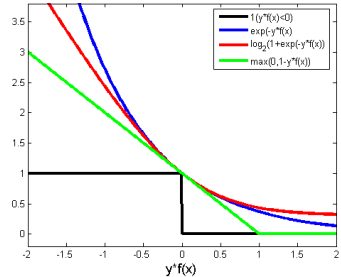
Empirical Risk Minimization

- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the average empirical loss

$$\widehat{f} = f_{\widehat{\theta}} = \underset{f_\theta, \theta \in \Theta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, f_\theta(\underline{x}_i))$$

- Plus convexification/regularization of the risk...
- Examples: SVM, (Deep) Neural Networks...

Classification Loss and Convexification



- Classification loss: $\ell^{0/1}(y, f(\underline{x})) = \mathbf{1}_{y \neq f(\underline{x})}$
- Not convex and not smooth!

Classical convexification

- Logistic loss: $\bar{\ell}(y, f(\underline{x})) = \log_2(1 + e^{-yf(\underline{x})})$ (Logistic / NN)
- Hinge loss: $\bar{\ell}(y, f(\underline{x})) = (1 - yf(\underline{x}))_+$ (SVM)
- Exponential loss: $\bar{\ell}(y, f(\underline{x})) = e^{-yf(\underline{x})}$ (Boosting...)

Properties

The Target is the Bayes Classifier

- The minimizer of
$$\mathbb{E}\Big[\bar{\ell}(Y, f(\underline{X}))\Big] = \mathbb{E}[I(Yf(\underline{X}))]$$
is the Bayes classifier $f^\star = \text{sign}(2\eta(\underline{X}) - 1)$

Control of the Excess Risk

- It exists a convex function Ψ such that
$$\Psi\left(\mathbb{E}\Big[\ell^{0/1}(Y, \text{sign}(f(\underline{X}))\Big] - \mathbb{E}\Big[\ell^{0/1}(Y, f^\star(\underline{X})\Big]\right)$$
$$\leq \mathbb{E}\Big[\bar{\ell}(Y, f(\underline{X})\Big] - \mathbb{E}\Big[\bar{\ell}(Y, f^\star(\underline{X}))\Big]$$

- Theoretical guarantee!

Logistic Revisited

- Ideal solution:

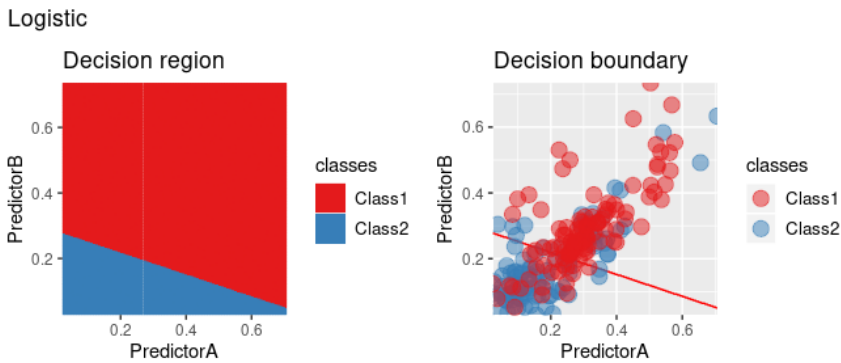
$$\widehat{f} = \underset{f \in \mathcal{S}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell^{0/1}(Y_i, f(\underline{X}_i))$$

Logistic regression

- Use $f(\underline{X}) = \underline{X}^\top \beta + \beta^{(0)}$.
- Use the logistic loss $\bar{\ell}(y, f) = \log_2(1 + e^{-yf})$, i.e. the negative log-likelihood.

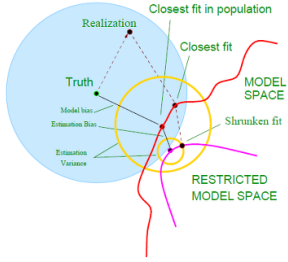
- Different vision than the statistician but same algorithm!

Logistic Revisited



Outline

Simplified Models



Bias-Variance Issue

- Most complex models may not be the best ones due to the variability of the estimate.

- Naive idea: can we *simplify* our model without loosing too much?
  - by using only a subset of the variables?
  - by forcing the coefficients to be small?
- Can we do better than exploring all possibilities?

Source: Tibshirani et al.

Linear Models

- **Setting**: Gen. linear model = prediction of Y by $h(\underline{x}^\top \beta)$.

Model coefficients

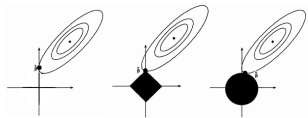
- Model entirely specified by β.
- Coefficientwise:
  - $\beta^{(i)} = 0$ means that the ith covariate is not used.
  - $\beta^{(i)} \sim 0$ means that the ith covariate as a *low* influence...

- If some covariates are useless, better use a simpler model...

Submodels

- *Simplify* the model through a constraint on β!
- Examples:
  - Support: Impose that $\beta^{(i)} = 0$ for $i \notin I$.
  - Support size: Impose that $\|\beta\|_0 = \sum_{i=1}^{d} \mathbf{1}_{\beta^{(i)} \neq 0} < C$
  - Norm: Impose that $\|\beta\|_p < C$ with $1 \leq p$ (Often $p = 2$ or $p = 1$)

Norms and Sparsity



Sparsity

- β is sparse if its number of non-zero coefficients (ℓ_0) is small...
- Easy interpretation in terms of dimension/complexity.

Norm Constraint and Sparsity

- Sparsest solution obtained by definition with the ℓ_0 norm.
- No induced sparsity with the ℓ_2 norm...
- Sparsity with the ℓ_1 norm (can even be proved to be the same as with the ℓ_0 norm under some assumptions).
- Geometric explanation.

Source: Tibshirani et al.

Constraint and Penalization

Constrained Optimization

- Choose a constant C.
- Compute β as

$$\underset{\beta \in \mathbb{R}^d, \|\beta\|_p \leq C}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \bar{\ell}(Y_i, h(\underline{x}_i{}^\top \beta))$$

Lagrangian Reformulation

- Choose λ and compute β as

$$\underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \bar{\ell}(Y_i, h(\underline{x}_i{}^\top \beta)) + \lambda \|\beta\|_p^{p'}$$

  with $p' = p$ except if $p = 0$ where $p' = 1$.
- Easier calibration... but no explicit model $\mathcal{S}$.

- **Rk:** $\|\beta\|_p$ is not scaling invariant if $p \neq 0$...
- Initial rescaling issue.

Penalization

Penalized Linear Model

- Minimization of

$$\underset{\beta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \bar{\ell}(Y_i, h(\underline{x}_i{}^\top \beta)) + \text{pen}(\beta)$$

  where $\text{pen}(\beta)$ is a (sparsity promoting) penalty
- Variable selection if β is sparse.

Classical Penalties

- AIC: $\text{pen}(\beta) = \lambda \|\beta\|_0$ (non-convex / sparsity)
- Ridge: $\text{pen}(\beta) = \lambda \|\beta\|_2^2$ (convex / no sparsity)
- Lasso: $\text{pen}(\beta) = \lambda \|\beta\|_1$ (convex / sparsity)
- Elastic net: $\text{pen}(\beta) = \lambda_1 \|\beta\|_1 + \lambda_2 \|\beta\|_2^2$ (convex / sparsity)

- Easy optimization if pen (and the loss) is convex...
- **Need to specify λ to define a ML method!**

Penalized Gen. Linear Models

Classical Examples

- Penalized Least Squares
- Penalized Logistic Regression
- Penalized Maximum Likelihood
- SVM
- Tree pruning

- Sometimes used even if the parameterization is not linear...

Practical Selection Methodology

- Choose a penalty family pen_λ.
- Compute a CV risk for the penalty pen_λ for all $\lambda \in \Lambda$.
- Determine $\widehat{\lambda}$ the λ minimizing the CV risk.
- Compute the final model with the penalty $\text{pen}_{\widehat{\lambda}}$.

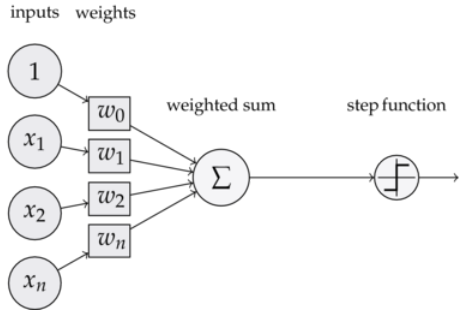
- CV allows to select a ML method, penalized estimation with a penalty $\text{pen}_{\widehat{\lambda}}$, not a single predictor hence the need of a final reestimation.

Why not using CV on a grid?

- Grid size scales exponentially with the dimension!
- **If the penalized minimization is easy**, much cheaper to compute the CV risk for all $\lambda \in \Lambda \ldots$
- CV performs best when the set of candidates is not too big (or is structured...)

Outline

Perceptron

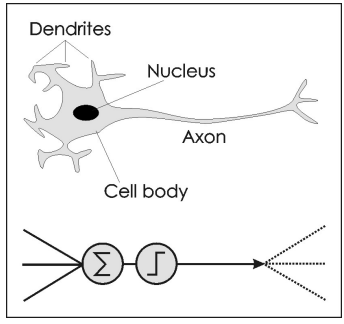


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

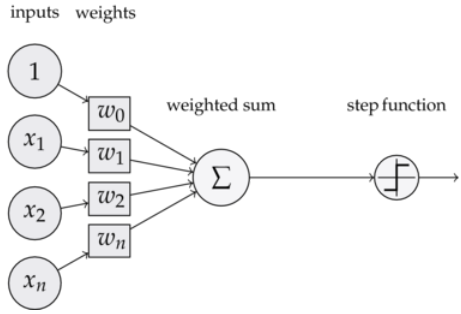


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

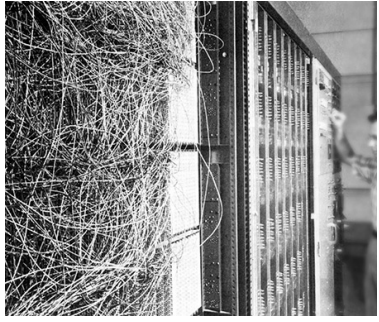


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Tikz

Perceptron

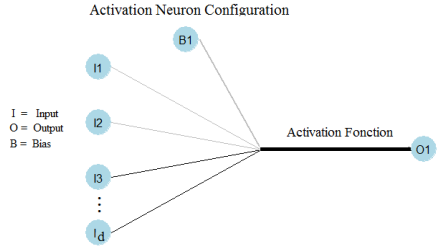


Perceptron (Rosenblatt 1957)

- Inspired from biology.
- Very simple (linear) model!
- Physical implementation and proof of concept.

Source: Avin Calspan Advanced Technology Center

Artificial Neuron and Logistic Regression



Artificial neuron

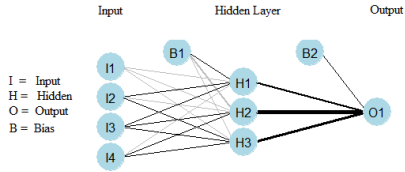
- Structure:
  - Mix inputs with a **weighted sum**,
  - Apply a (non linear) **activation function** to this sum,
  - Possibly threshold the result to make a decision.
- Weights learned by minimizing a loss function.

Logistic unit

- Structure:
  - Mix inputs with a **weighted sum**,
  - Apply the **logistic function** $\sigma(t) = e^t/(1+e^t)$,
  - Threshold at $1/2$ to make a decision!
- Logistic weights learned by minimizing the -log-likelihood.

Source: Unknown

Multilayer Perceptron



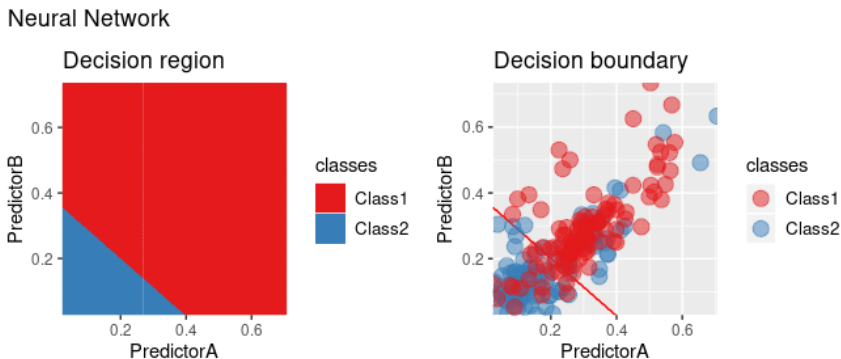
MLP (Rumelhart, McClelland, Hinton - 1986)

- Multilayer Perceptron: cascade of layers of artificial neuron units.
- Optimization through a gradient descent algorithm with a clever implementation (**Backprop**).

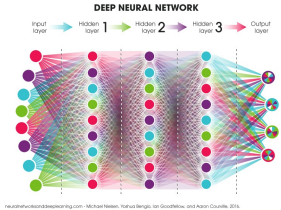
- Construction of a function by composing simple units.
- MLP corresponds to a specific direct acyclic graph structure.
- Non convex optimization problem!

Source: Unknown

Multilayer Perceptron



Deep Neural Network



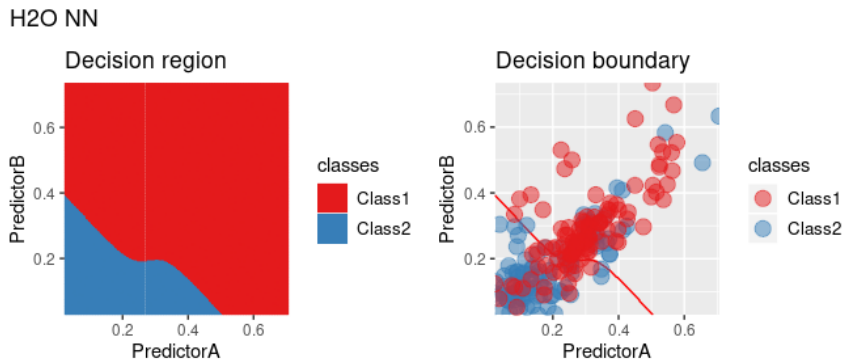
Deep Neural Network structure

- Deep cascade of layers!

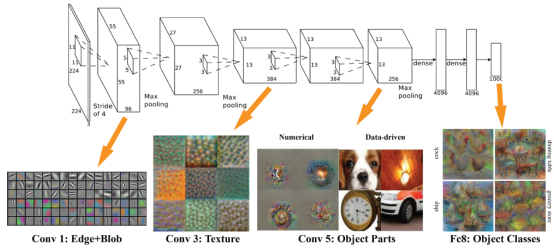
- No conceptual novelty...
- But a **lot of tricks** allowing to obtain a good solution: clever initialization, better activation function, weight regularization, accelerated stochastic gradient descent, early stopping...
- Use of GPU and a lot of data...
- Very impressive results!

Source: Nielsen, Bengio, Goodfellow and Courville

Deep Neural Network



Deep Learning



Family of Machine Learning algorithm combining:

- a (deep) multilayered structure,
- a clever optimization including initialization and regularization.

- Examples: Deep NN, AutoEncoder, Recursive NN, GAN, Transformer...
- Interpretation as a **Representation Learning**.
- **Transfer learning:** use as initialization a pretrained net.
- Very efficient and still evolving!

Source: J. Hays

Outline

Support Vector Machine

$$f_\theta(\underline{X}) = \underline{X}^\top \beta + \beta^{(0)} \quad \text{with} \quad \theta = (\beta, \beta^{(0)})$$

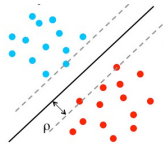
$$\hat{\theta} = \arg\min \frac{1}{n} \sum_{i=1}^{n} \max\left(1 - Y_i f_\theta(\underline{X}_i), 0\right) + \lambda \|\beta\|_2^2$$

Support Vector Machine

- Convexification of the 0/1-loss with the hinge loss:
$$\mathbf{1}_{Y_i f_\theta(\underline{X}_i) < 0} \leq \max\left(1 - Y_i f_\theta(\underline{X}_i), 0\right)$$
- Penalization by the quadratic norm (Ridge/Tikhonov).
- Solution can be approximated by gradient descent algorithms.

- **Revisit** of the original point of view.
- Original point of view leads to a different optimization algorithm and to some extensions.

Ideal Separable Case



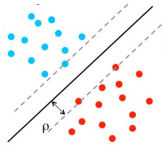
- Linear classifier: $\text{sign}(\underline{X}^\top \beta + \beta^{(0)})$
- Separable case: $\exists(\beta, \beta^{(0)}), \forall i,\ Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) > 0$

How to choose $(\beta, \beta^{(0)})$ so that the separation is maximal?

- Strict separation: $\exists(\beta, \beta^{(0)}), \forall i,\ Y_i(\underline{X_i}^\top \beta + \beta^{(0)}) \geq 1$
- Distance between $\underline{X}^\top \beta + \beta^{(0)} = 1$ and $\underline{X}^\top \beta + \beta^{(0)} = -1$:
$$\frac{2}{\|\beta\|}$$
- Maximizing this distance is equivalent to minimizing $\frac{1}{2}\|\beta\|^2$.

Source: M. Mohri et al.

Ideal Separable Case



Separable SVM

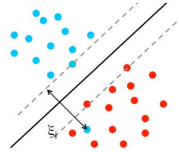
- Constrained optimization formulation:

$$\min \frac{1}{2}\|\beta\|^2 \quad \text{with} \quad \forall i, Y_i(\underline{X}_i^\top \beta + \beta^{(0)}) \geq 1$$

- Quadratic Programming setting.
- Efficient solver available...

Source: M. Mohri et al.

Non Separable Case



- What about the non separable case?

SVM relaxation

- Relax the assumptions

$$\forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1 \quad \text{to} \quad \forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1 - s_i$$

with the **slack variables** $s_i \geq 0$

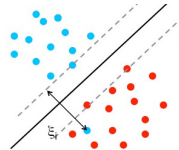
- Keep those slack variables as small as possible by minimizing

$$\frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i$$

where $C > 0$ is the **goodness-of-fit strength**

Source: M. Mohri et al.

Non Separable Case



SVM

- Constrained optimization formulation:

$$\min \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} s_i \quad \text{with} \quad \begin{cases} \forall i, Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}) \geq 1 - s_i \\ \forall i, s_i \geq 0 \end{cases}$$

- **Hinge Loss** reformulation:

$$\min \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \underbrace{\max(0, 1 - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}))}_{\text{Hinge Loss}}$$

- Constrained convex optimization algorithms vs gradient descent algorithms.

Source: M. Mohri et al.

SVM as a Penalized Convex Relaxation

- Convex relaxation:

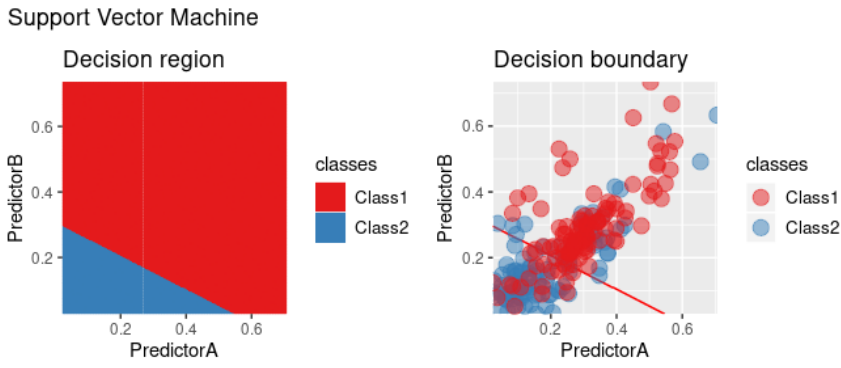
$$\begin{aligned}&\operatorname{argmin} \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{n} \max(1 - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}), 0)\\ &= \operatorname{argmin} \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}), 0) + \frac{1}{Cn}\frac{1}{2}\|\beta\|^2\end{aligned}$$

- **Prop:** $\ell^{0/1}(Y_i, \operatorname{sign}(\underline{X}_i{}^\top \beta + \beta^{(0)})) \leq \max(1 - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}), 0)$

Penalized convex relaxation (Tikhonov!)

$$\begin{aligned}&\frac{1}{n}\sum_{i=1}^{n} \ell^{0/1}(Y_i, \operatorname{sign}(\underline{X}_i{}^\top \beta + \beta^{(0)})) + \frac{1}{Cn}\frac{1}{2}\|\beta\|^2\\ &\leq \frac{1}{n}\sum_{i=1}^{n} \max(1 - Y_i(\underline{X}_i{}^\top \beta + \beta^{(0)}), 0) + \frac{1}{Cn}\frac{1}{2}\|\beta\|^2\end{aligned}$$

SVM


Support Vector Machine
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Mercer Theorem and Scalar Product

- **Mercer Theorem:** the minimizer in β of

$$\frac{1}{n}\sum_{i=1}^{n}\max(1-Y_i(\underline{X}_i^\top\beta+\beta^{(0)}),0)+\frac{1}{Cn}\frac{1}{2}\|\beta\|^2$$

  is a linear combination of the input points $\sum_{i=1}^{n}\alpha_i'\underline{X}_i$.
- **Duality theory:** $\alpha_i'=\alpha_i Y_i$ where

$$\alpha=\arg\max\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j Y_i Y_j\underline{X}_i^\top\underline{X}_j$$

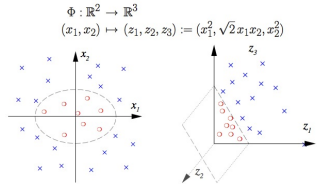
  under the constraints $\sum_{i=1}^{n}\alpha_i Y_i=0$ and $0\leq\alpha_i\leq C$.

Dual formulation

- α_i are **Lagrangian multipliers** and are equal to 0 as soon as $y_i(\underline{X}_i^\top\beta+\beta^{(0)})>1$
- Explicit formula for $\beta^{(0)}$.
- Data involved only through **scalar product** $\underline{X}^\top\underline{X}'$!
- Quadratic programming reformulation!

- **Suport Vectors** are the ones for which $\alpha_i\neq 0$.

The Kernel Trick



- Non linear separation: just replace $\underline{X}$ by a non linear $\Phi(\underline{X})\ldots$
- Knowing $\phi(\underline{X}_i)^\top \phi(\underline{X}_j)$ is sufficient to compute the SVM solution.

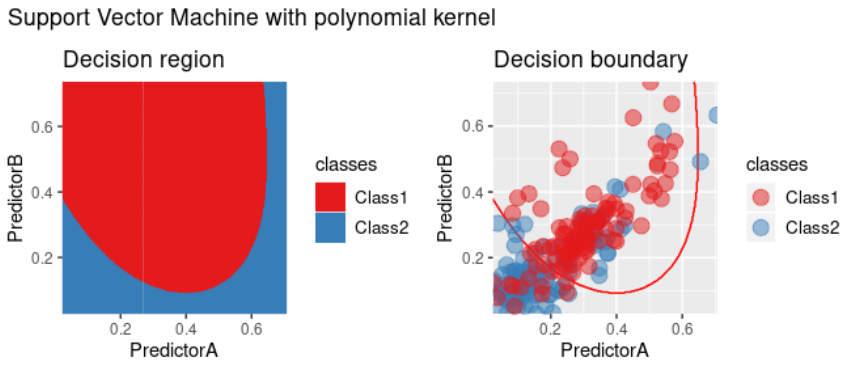
Kernel trick

- **Computing $k(\underline{X}, \underline{X}') = \phi(\underline{X})^\top \phi(\underline{X}')$ may be easier than computing $\phi(\underline{X})$, $\phi(\underline{X}')$ and then the scalar product!**
- ϕ can be specified through its definite positive kernel k.

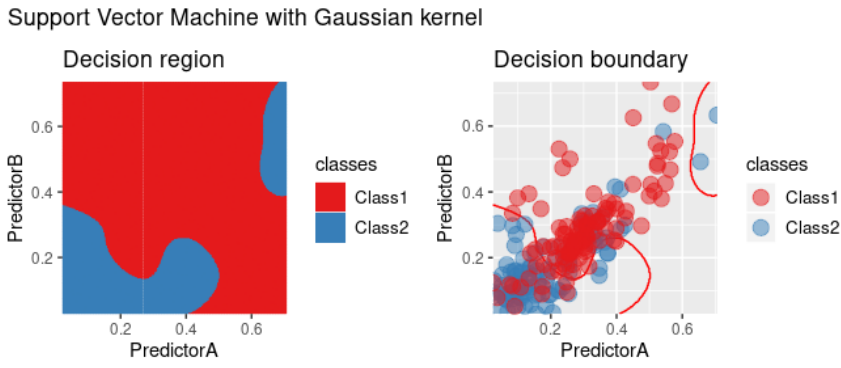
- Examples: Polynomial kernel $k(\underline{X}, \underline{X}') = (1 + \underline{X}^\top \underline{X}')^d$, Gaussian kernel $k(\underline{X}, \underline{X}') = e^{-\|\underline{X}-\underline{X}'\|^2/2}$,...
- RKHS setting!
- Can be used in (logistic) regression and more...

Source: Unknown

SVM


Support Vector Machine with polynomial kernel
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

SVM


Support Vector Machine with Gaussian kernel
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Outline

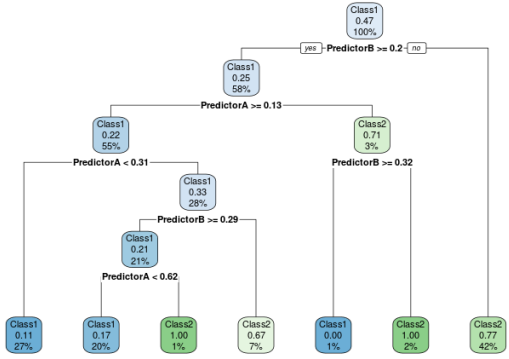
Classification Trees

Tree principle (CART by Breiman (85) / ID3 by Quinlan (86))

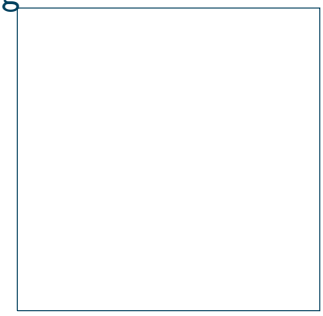
- Construction of a recursive partition through a tree structured set of questions (splits around a given value of a variable)
- For a given partition, probabilistic approach **and** optimization approach yield the same predictor!
- A simple majority vote in each leaf

- Quality of the prediction depends on the tree (the partition).
- **Intuitively:**
  - small leaves lead to low bias, but large variance
  - large leaves lead to large bias, but low variance...
- **Issue:** Minim. of the (penalized) empirical risk is NP hard!
- Practical tree construction are all based on two steps:
  - a top-down step in which branches are created (branching)
  - a bottom-up in which branches are removed (pruning)

CART


Class1
0.47
100%
yes
PredictorB >= 0.2
no
Class1
0.25
58%
PredictorA >= 0.13
Class1
0.22
55%
Class2
0.71
3%
PredictorA < 0.31
PredictorB >= 0.32
Class1
0.33
28%
PredictorB >= 0.29
Class1
0.21
21%
PredictorA < 0.62
Class1
0.11
27%
Class1
0.17
20%
Class2
1.00
1%
Class2
0.67
7%
Class1
0.00
1%
Class2
1.00
2%
Class2
0.77
42%

Branching

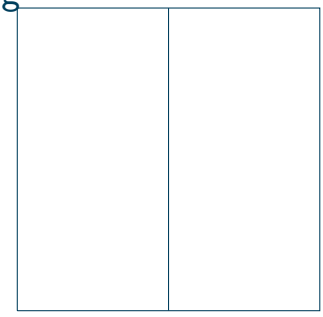


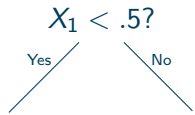
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



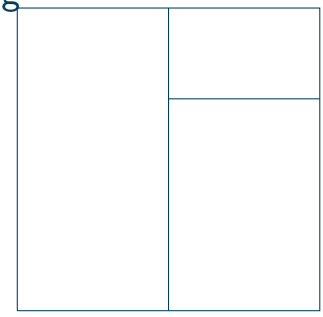


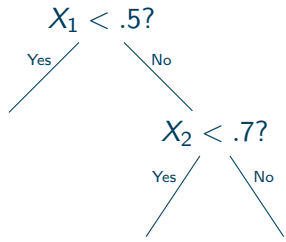
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



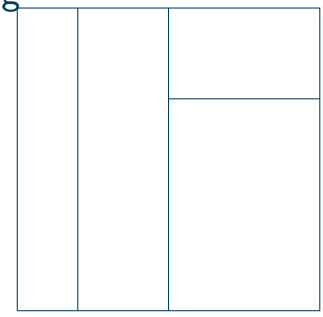


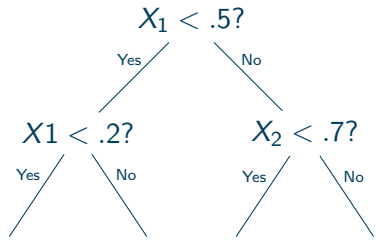
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching





Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible. . .

Branching

Various definition of in*homogeneous*

- **CART:** empirical loss based criterion (least squares/prediction error)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} \bar{\ell}(y_i, y(R)) + \sum_{\underline{x}_i \in \overline{R}} \bar{\ell}(y_i, y(\overline{R}))$$
- **CART:** Gini index (Classification)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} p(R)(1 - p(R)) + \sum_{\underline{x}_i \in \overline{R}} p(\overline{R})(1 - p(\overline{R}))$$
- **C4.5:** entropy based criterion (Information Theory)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} H(R) + \sum_{\underline{x}_i \in \overline{R}} H(\overline{R})$$

- CART with Gini is probably the most used technique...
- Other criterion based on χ^2 homogeneity or based on different local predictors (generalized linear models...)

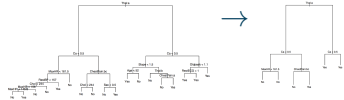
Branching

Choice of the split in a given region

- Compute the criterion for **all features and all possible splitting points** (necessarily among the data values in the region)
- Choose the split **minimizing** the criterion

- **Variations:** split at all categories of a categorical variable using a clever category ordering (ID3), split at a restricted set of points (quantiles or fixed grid)
- **Stopping rules:**
  - when a leaf/region contains less than a prescribed number of observations
  - when the region is sufficiently homogeneous...
- May lead to a quite complex tree: over-fitting possible!
- Additional pruning often use.

Pruning



- **Model selection** within the (rooted) subtrees of previous tree!
- Number of subtrees can be quite large, but the tree structure allows to find the best model efficiently.

Key idea

- The predictor in a leaf depends only on the values in this leaf.
- **Efficient bottom-up (dynamic programming) algorithm** if the criterion used satisfies an additive property

$$C(\mathcal{T}) = \sum_{\mathcal{L} \in \mathcal{T}} c(\mathcal{L})$$

- Example: AIC / CV.

Examples of criterion satisfying this assumptions

- AIC type criterion:

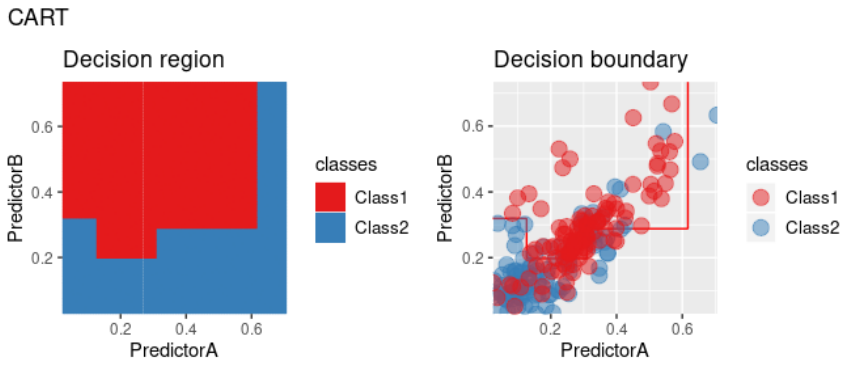
$$\sum_{i=1}^{n} \bar{\ell}(y_i, f_{\mathcal{L}(\underline{x}_i)}(\underline{x}_i)) + \lambda|\mathcal{T}| = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}_i \in \mathcal{L}} \bar{\ell}(y_i, f_{\mathcal{L}}(\underline{x}_i)) + \lambda \right)$$

- Simple cross-Validation (with $(\underline{x}'_i, y'_i)$ a different dataset):

$$\sum_{i=1}^{n'} \bar{\ell}(y'_i, f_{\mathcal{L}}(\underline{x}'_i)) = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}'_i \in \mathcal{L}} \bar{\ell}(y'_i, f_{\mathcal{L}}(\underline{x}'_i)) \right)$$

- Limit over-fitting for a single tree.
- **Rk:** almost never used when combining several trees...

CART


CART
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

CART: Pros and Cons

Pros

- Leads to an easily interpretable model
- Fast computation of the prediction
- Easily deals with categorical features (and missing values)

Cons

- Greedy optimization
- Hard decision boundaries
- Lack of stability

Ensemble methods

- Lack of robustness for single trees.
- How to combine trees?

Parallel construction

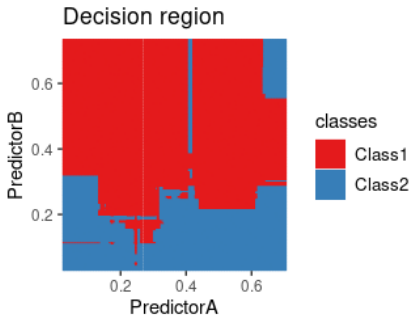
- Construct several trees from bootstrapped samples and average the responses **(Bagging)**
- Add more randomness in the tree construction **(Random Forests)**

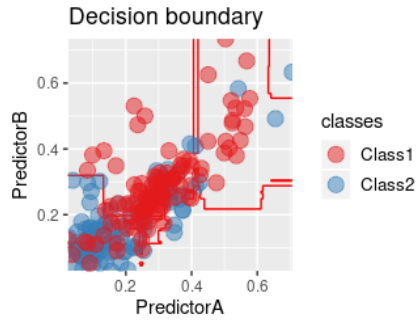
Sequential construction

- Construct a sequence of trees by reweighting sequentially the samples according to their difficulties **(AdaBoost)**
- Reinterpretation as a stagewise additive model **(Boosting)**

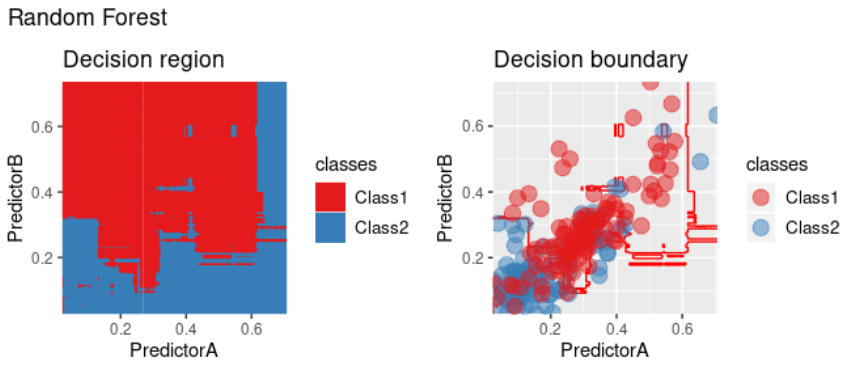
Ensemble methods

Bagging

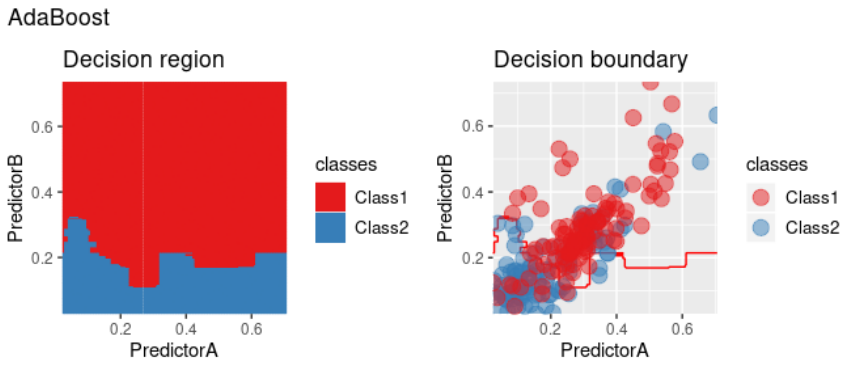




Ensemble methods



Ensemble methods



Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

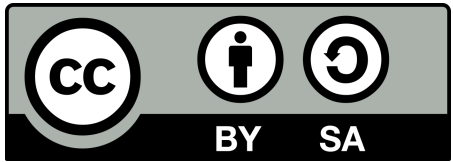
K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

Outline

Guess Who?

A game of questions

- Game invented in 1979 in the UK.
- **Goal:** discover the character chosen by your opponent before he discovers yours.
- **Optimal strategy:** choose at each step the question that splits the remaining characters in two groups with the least possible difference in size.
- **Information Theory!**

- Adaptive construction of a tree of questions!
- Optimal tree of questions can be constructed without knowing the answers... but during a game only a path of the tree is used...

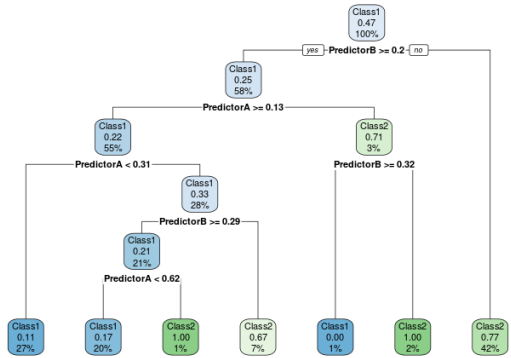
Classification And Regression Trees

Tree principle (CART by Breiman (85) / ID3 by Quinlan (86))

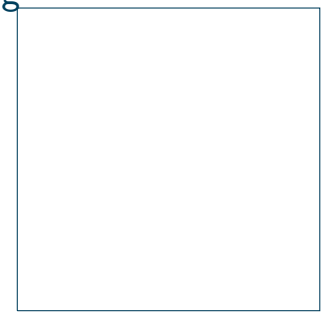
- Construction of a recursive partition through a tree structured set of questions (splits around a given value of a variable)
- For a given partition, probabilistic approach **and** optimization approach yield the same predictor!
- A simple majority vote/averaging in each leaf

- Quality of the prediction depends on the tree (the partition).
- **Intuitively:**
  - small leaves lead to low bias, but large variance
  - large leaves lead to large bias, but low variance...
- **Issue:** Minim. of the (penalized) empirical risk is NP hard!
- Practical tree construction are all based on two steps:
  - a top-down step in which branches are created (branching)
  - a bottom-up in which branches are removed (pruning)

CART


Class1
0.47
100%
yes
PredictorB >= 0.2
no
Class1
0.25
58%
PredictorA >= 0.13
Class1
0.22
55%
Class2
0.71
3%
PredictorA < 0.31
PredictorB >= 0.32
Class1
0.33
28%
PredictorB >= 0.29
Class1
0.21
21%
PredictorA < 0.62
Class1
0.11
27%
Class1
0.17
20%
Class2
1.00
1%
Class2
0.67
7%
Class1
0.00
1%
Class2
1.00
2%
Class2
0.77
42%

Branching

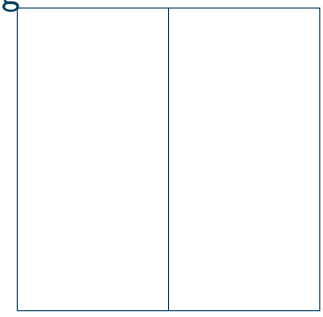


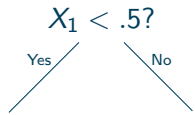
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



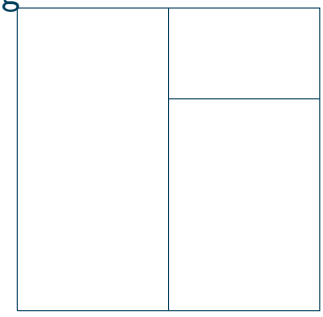


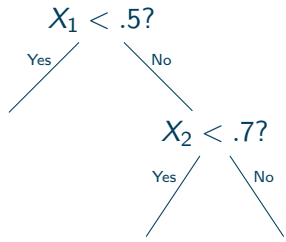
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching



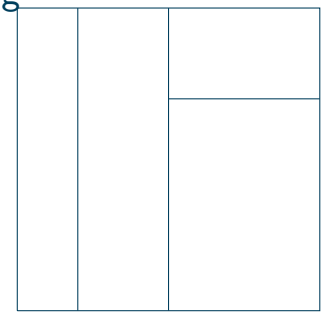


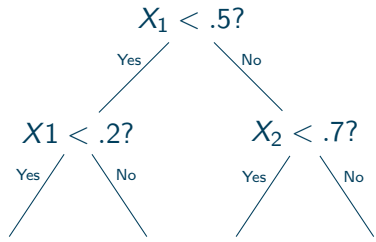
Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching





Greedy top-bottom approach

- Start from a single region containing all the data
- Recursively split those regions along a certain variable and a certain value

- **No regret strategy** on the choice of the splits!
- **Heuristic:** choose a split so that the two new regions are as *homogeneous* possible...

Branching

Various definition of in*homogeneous*

- **CART:** empirical loss based criterion (least squares/prediction error)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} \bar{\ell}(y_i, y(R)) + \sum_{\underline{x}_i \in \overline{R}} \bar{\ell}(y_i, y(\overline{R}))$$
- **CART:** Gini index (Classification)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} p(R)(1 - p(R)) + \sum_{\underline{x}_i \in \overline{R}} p(\overline{R})(1 - p(\overline{R}))$$
- **C4.5:** entropy based criterion (Information Theory)
$$C(R, \overline{R}) = \sum_{\underline{x}_i \in R} H(R) + \sum_{\underline{x}_i \in \overline{R}} H(\overline{R})$$

- CART with Gini is probably the most used technique...
- Other criterion based on χ^2 homogeneity or based on different local predictors (generalized linear models...)

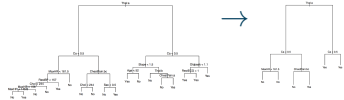
Branching

Choice of the split in a given region

- Compute the criterion for **all features and all possible splitting points** (necessarily among the data values in the region)
- Choose the split **minimizing** the criterion

- **Variations:** split at all categories of a categorical variable using a clever category ordering (ID3), split at a restricted set of points (quantiles or fixed grid)
- **Stopping rules:**
  - when a leaf/region contains less than a prescribed number of observations
  - when the region is sufficiently homogeneous...
- May lead to a quite complex tree: over-fitting possible!
- Additional pruning often use.

Pruning



- **Model selection** within the (rooted) subtrees of previous tree!
- Number of subtrees can be quite large, but the tree structure allows to find the best model efficiently.

Key idea

- The predictor in a leaf depends only on the values in this leaf.
- **Efficient bottom-up (dynamic programming) algorithm** if the criterion used satisfies an additive property

$$C(\mathcal{T}) = \sum_{\mathcal{L} \in \mathcal{T}} c(\mathcal{L})$$

- Example: AIC / CV.

Pruning

Examples of criterion satisfying this assumptions

- AIC type criterion:

$$\sum_{i=1}^{n} \bar{\ell}(y_i, f_{\mathcal{L}(\underline{x}_i)}(\underline{x}_i)) + \lambda|\mathcal{T}| = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}_i \in \mathcal{L}} \bar{\ell}(y_i, f_{\mathcal{L}}(\underline{x}_i)) + \lambda \right)$$

- Simple cross-Validation (with $(\underline{x}'_i, y'_i)$ a different dataset):

$$\sum_{i=1}^{n'} \bar{\ell}(y'_i, f_{\mathcal{L}}(\underline{x}'_i)) = \sum_{\mathcal{L}\in\mathcal{T}} \left(\sum_{\underline{x}'_i \in \mathcal{L}} \bar{\ell}(y'_i, f_{\mathcal{L}}(\underline{x}'_i)) \right)$$

- Limit over-fitting for a single tree.
- **Rk:** almost never used when combining several trees...

Pruning and Dynamic Algorithm

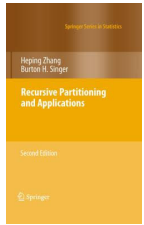
- **Key observation:** at a given node, the best subtree is either the current node or the union of the best subtrees of its child.

Dynamic programming algorithm

- Compute the individual cost $c(\mathcal{L})$ of each node (including the leaves)
- Scan all the nodes in reverse order of depth:
  - If the node $\mathcal{L}$ has no child, set its best subtree $\mathcal{T}(\mathcal{L})$ to $\{\mathcal{L}\}$ and its current best cost $c'(\mathcal{L})$ to $c(\mathcal{L})$
  - If the children $\mathcal{L}_1$ and $\mathcal{L}_2$ are such that $c'(\mathcal{L}_1) + c'(\mathcal{L}_2) \geq c(\mathcal{L})$, then prune the child by setting $\mathcal{T}(\mathcal{L}) = \{\mathcal{L}\}$ and $c'(\mathcal{L}) = c(\mathcal{L})$
  - Otherwise, set $\mathcal{T}(\mathcal{L}) = \mathcal{T}(\mathcal{L}_1) \cup \mathcal{T}(\mathcal{L}_2)$ and $c'(\mathcal{L}) = c'(\mathcal{L}_1) + c'(\mathcal{L}_2)$
- The best subtree is the best subtree $\mathcal{T}(\mathcal{R})$ of the root $\mathcal{R}$.

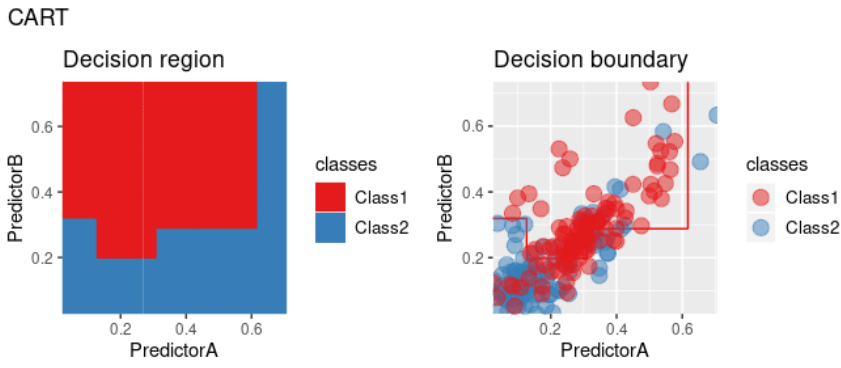
- Optimization cost proportional to the **number of nodes** and not the number of subtrees!

Extensions



- **Local estimation** of the proportions or of the conditional mean.
- **Recursive Partitioning methods:**
  - Recursive construction of a partition
  - Use of simple local model on each part of the partition
- **Examples:**
  - CART, ID3, C4.5, C5
  - MARS (local linear regression models)
  - Piecewise polynomial model with a dyadic partition...
- **Book:** *Recursive Partitioning and Applications* by Zhang and Singer

CART


CART
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

CART: Pros and Cons

Pros

- Leads to an easily interpretable model
- Fast computation of the prediction
- Easily deals with categorical features (and missing values)

Cons

- Greedy optimization
- Hard decision boundaries
- Lack of stability

Ensemble methods

- Lack of robustness for single trees.
- How to combine trees?

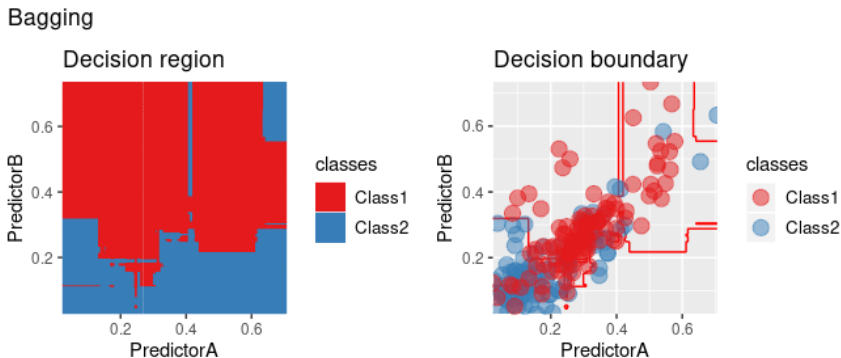
Parallel construction

- Construct several trees from bootstrapped samples and average the responses **(Bagging)**
- Add more randomness in the tree construction **(Random Forests)**

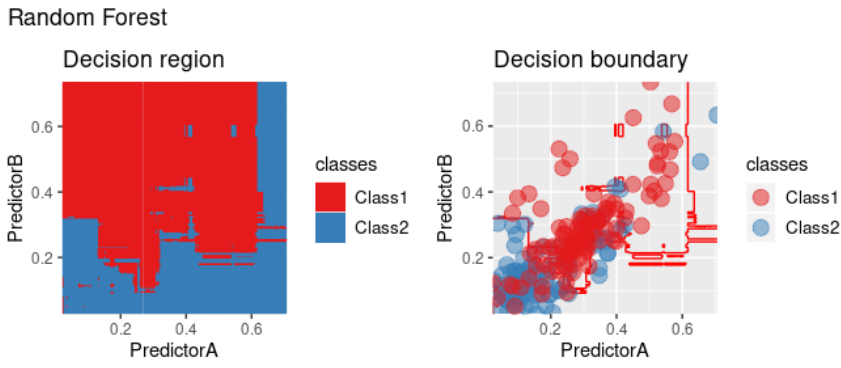
Sequential construction

- Construct a sequence of trees by reweighting sequentially the samples according to their difficulties **(AdaBoost)**
- Reinterpretation as a stagewise additive model **(Boosting)**

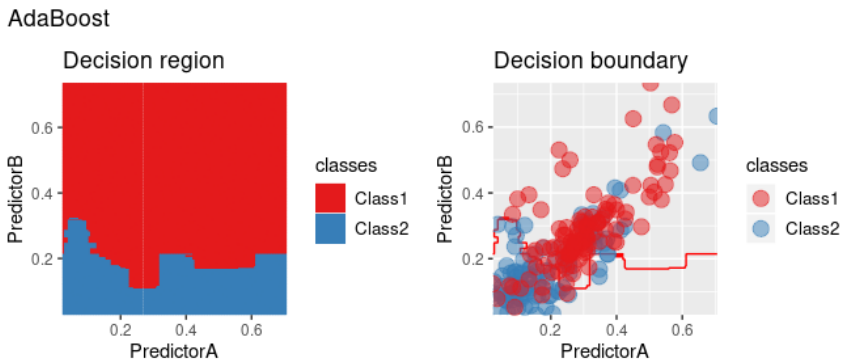
Ensemble methods



Ensemble methods



Ensemble methods



Outline

Outline

Independent Average

Stability through averaging

- Very simple idea to obtain a more stable estimator.
- **Vote/average** of B predictors $f_1, \ldots, f_B$ obtained with **independent datasets** of size n!

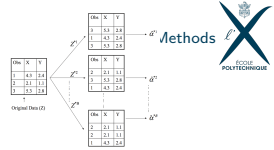
$$f_{\text{agr}} = \text{sign}\left(\frac{1}{B}\sum_{b=1}^{B} f_b\right) \quad \text{or} \quad f_{\text{agr}} = \frac{1}{B}\sum_{i=1}^{B} f_b$$

- **Regression:** $\mathbb{E}[f_{\text{agr}}(x)] = \mathbb{E}[f_b(x)]$ and $\mathbb{V}\text{ar}\left[f_{\text{agr}}(x)\right] = \frac{\mathbb{V}\text{ar}[f_b(x)]}{B}$
- **Prediction:** slightly more complex analysis
- Averaging leads to **variance reduction**, i.e. stability!

- **Issue:** cost of obtaining B independent datasets of size n!

Bagging and Bootstrap

- Strategy proposed by Breiman in 1994.



Stability through bootstrapping

- Instead of using B independent datasets of size n, draw B datasets from a single one using a **uniform with replacement** scheme (Bootstrap).
- **Rk:** On average, a fraction of $(1 - 1/e) \simeq .63$ examples are unique among each drawn dataset...
- The f_b are still identically distributed but **not independent** anymore.
- Price for the non independence: $\mathbb{E}[f_{\text{agr}}(x)] = \mathbb{E}[f_b(x)]$ and

$$\mathbb{V}\text{ar}\left[f_{\text{agr}}(x)\right] = \frac{\mathbb{V}\text{ar}\left[f_b(x)\right]}{B} + \left(1 - \frac{1}{B}\right)\rho(x)$$

  with $\rho(x) = \mathbb{C}\text{ov}\left[f_b(x), f_{b'}(x)\right] \leq \mathbb{V}\text{ar}\left[f_b(x)\right]$ with $b \neq b'$.
- **Bagging:** Bootstrap Aggregation

- Better aggregation scheme exists...

Outline

Randomized Predictors

- Correlation leads to less variance reduction:
$$\mathbb{V}\mathrm{ar}\left[f_{\mathrm{agr}}(x)\right] = \frac{\mathbb{V}\mathrm{ar}\left[f_b(x)\right]}{B} + \left(1 - \frac{1}{B}\right)\rho(x)$$
with $\rho(x) = \mathbb{C}\mathrm{ov}\left[f_b(x), f_{b'}(x)\right]$ with $b \neq b'$.
- **Idea:** Reduce the correlation by adding more randomness in the predictor.

Randomized Predictors

- Construct predictors that depend on a **randomness source** R that may be chosen independently for all bootstrap samples.
- This **reduces** the correlation between the estimates and thus the **variance**...
- But may **modify heavily the estimates** themselves!

- **Performance gain** not obvious from theory...

Random Forest

- Example of randomized predictors based on trees proposed by Breiman in 2001...

Random Forest

- Draw B resampled datasets from a single one using a uniform with replacement scheme (**Bootstrap**)
- For each resampled dataset, construct a tree using a different **randomly drawn subset of variables** at each split.

- Most important parameter is the **subset size**:
  - if it is too large then we are back to bagging
  - if it is too small the mean of the predictors is probably not a good predictor...
- **Recommendation:**
  - Classification: use a proportion of $1/\sqrt{p}$
  - Regression: use a proportion of $1/3$
- **Sloppier stopping rules** and pruning than in CART...

Extra Trees

- Extremely randomized trees!

Extra Trees

- Variation of random forests.
- Instead of trying all possible cuts, try only K cuts at random for each variable.
- No bootstrap in the original article.

- Cuts are defined by a threshold drawn uniformly in the feature range.
- Much faster than the original forest and similar performance.
- Theoretical performance analysis very challenging!

Error Estimate and Variable Ranking

Out Of the Box Estimate

- For each sample x_i, a prediction can be made using only the resampled datasets not containing x_i...
- The corresponding empirical prediction error is **not prone to overfitting** but does not correspond to the final estimate...
- Good proxy nevertheless.

Forests and Variable Ranking

- **Importance:** Number of use or criterion gain at each split can be used to rank the variables.
- **Permutation tests:** Difference between OOB estimate using the true value of the jth feature and a value drawn a random from the list of possible values.

- Up to OOB error, the permutation technique is not specific to trees.

Outline

Outline

AdaBoost

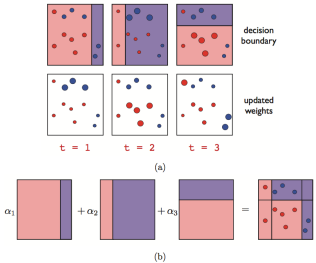
- **Idea:** learn a sequence of predictors trained on weighted dataset with weights depending on the loss so far.

Iterative scheme proposed by Schapire and Freud

- Set $w_{1,i} = 1/n$; $t = 0$ and $f = 0$
- For $t = 1$ to $t = T$
  - $h_t = \text{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{n} w_{t,i} \ell^{0/1}(y_i, h(x_i))$
  - Set $\epsilon_t = \sum_{i=1}^{n} w_{t,i} \ell^{0/1}(y_i, h_t(x_i))$ and $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
  - let $w_{t+1,i} = \frac{w_{t,i} e^{-\alpha_t y_i h_t(x_i)}}{Z_{t+1}}$ where Z_{t+1} is a renormalization constant such that $\sum_{i=1}^{n} w_{t+1,i} = 1$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{i=1}^{T} \alpha_t h_t$

- **Intuition:** $w_{t,i}$ measures the difficulty of learning the sample i at step t...

AdaBoost



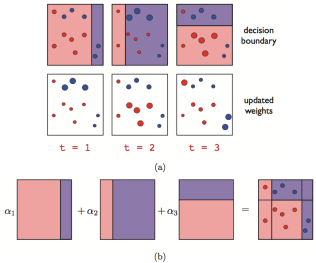
AdaBoost Intuition

- h_t obtained by minimizing a weighted loss

$$h_t = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} w_{t,i} \ell^{0/1}(y_i, h(\underline{x}_i))$$

- Update the current estimate with

$$f_t = f_{t-1} + \alpha_t h_t$$

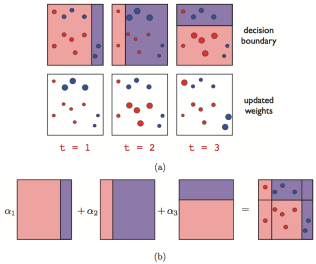


AdaBoost Intuition

- Weight $w_{t,i}$ should be large if $\underline{x}_i$ is not well-fitted at step $t-1$ and small otherwise.
- Use a weight proportional to $e^{-y_i f_{t-1}(\underline{x}_i)}$ so that it can be recursively updated by

$$w_{t+1,i} = w_{t,i} \times \frac{e^{-\alpha_t y_i h_t(\underline{x}_i)}}{Z_t}$$

AdaBoost



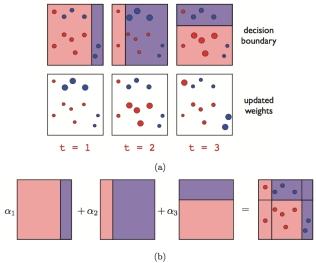
AdaBoost Intuition

- Set α_t such that

$$\sum_{y_i h_t(\underline{x}i)=1} w_{t+1,i} = \sum_{y_i h_t(\underline{x}i)=-1} w_{t+1,i}$$

or equivalently

$$\left(\sum_{y_i h_t(\underline{x}i)=1} w_{t,i} \right) e^{-\alpha_t} = \left(\sum_{y_i h_t(\underline{x}i)=-1} w_{t,i} \right) e^{\alpha_t}$$



AdaBoost Intuition

- Using

$$\epsilon_t = \sum_{y_i h_t(\underline{x} i)=-1} w_{t,i}$$

leads to

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} \quad \text{and} \quad Z_t = 2\sqrt{\epsilon_t(1-\epsilon_t)}$$

Exponential Stagewise Additive Modeling

- Set $t = 0$ and $f = 0$.
- For $t = 1$ to T,
  - $(h_t, \alpha_t) = \text{argmin}_{h,\alpha} \sum_{i=1}^{n} e^{-y_i(f(\underline{x}_i)+\alpha h(\underline{x}_i))}$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{t=1}^{T} \alpha_t h_t$

- **Greedy optimization** of a classifier as a linear combination of T classifier for the **exponential loss**.
- Those two algorithms are **exactly the same**!

Revisited AdaBoost

AdaBoost

- Set $t = 0$ and $f = 0$.
- For $t = 1$ to T,
  - $(h_t, \alpha_t) = \text{argmin}_{h,\alpha} \sum_{i=1}^{n} e^{-y_i(f(\underline{x}_i)+\alpha h(\underline{x}_i))}$
  - $f = f + \alpha_t h_t$
- Use $f = \sum_{t=1}^{T} \alpha_t h_t$

- **Greedy iterative scheme** with only two parameters: the class $\mathcal{H}$ of *weak* classifiers and the number of step T.
- In the literature, one can read that Adaboost does not overfit! This is not true and T should be chosen with care...

Outline

Weak Learners

Weak Learner

- Simple predictor belonging to a set $\mathcal{H}$.
- Easy to learn.
- Need to be only slightly better than a constant predictor.

Weak Learner Examples

- **Decision Tree** with few splits.
- **Stump** decision tree with one split.
- **(Generalized) Linear Regression** with few variables.

Boosting

- Sequential Linear Combination of Weak Learner
- Attempt to minimize a loss.

- Example of ensemble method.
- Link with Generalized Additive Modeling.

Generic Boosting

- **Greedy optim.** yielding a linear combination of *weak* learners.

Generic Boosting

- Algorithm:
  - Set $t = 0$ and $f = 0$.
  - For $t = 1$ to T,
    - $(h_t, \alpha_t) = \text{argmin}_{h,\alpha} \sum_{i=1}^{n} \bar{\ell}(y_i, f(x_i) + \alpha h(x_i))$
    - $f = f + \alpha_t h_t$
  - Use $f = \sum_{t=1}^{T} \alpha_t h_t$
- AKA as **Forward Stagewise Additive Modeling**
  - AdaBoost with $\bar{\ell}(y, h) = e^{-yh}$
  - LogitBoost with $\bar{\ell}(y, h) = \log_2(1 + e^{-yh})$
  - L_2Boost with $\bar{\ell}(y, h) = (y - h)^2$ (Matching pursuit)
  - L_1Boost with $\bar{\ell}(y, h) = |y - h|$
  - HuberBoost with $\bar{\ell}(y, h) = |y - h|^2 \mathbf{1}_{|y-h|<\epsilon} + (2\epsilon|y - h| - \epsilon^2)\mathbf{1}_{|y-h|\geq\epsilon}$

- Simple principle but **no easy numerical scheme** except for AdaBoost and L_2Boost...

Gradient Boosting

- **Issue:** At each boosting step, one need to solve

$$(h_t, \alpha_t) = \underset{h,\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} \bar{\ell}(y_i, f(x_i) + \alpha h(x_i)) = L(y, f + \alpha h)$$

- **Idea:** Replace the function by a **first order approximation**

$$L(y, f + \alpha h) \sim L(y, f) + \alpha \langle \nabla L(y, f), h \rangle$$

Gradient Boosting

- Replace the minimization step by a **gradient descent** step:
  - Choose h_t as the best possible descent direction in $\mathcal{H}$ according to the approximation
  - Choose α_t that minimizes $L(y, f + \alpha h_t)$ (line search)

- **Rk:** Exact gradient direction often not possible!
- Need to find efficiently this best possible direction...

Best Direction

- Gradient direction:

$$\nabla L(y,f) \quad \text{with} \quad \nabla_i L(y,f) = \frac{\partial}{df(x_i)} \left(\sum_{i'=1}^{n} \bar{\ell}(y_{i'}, f(x_{i'})) \right)$$
$$= \frac{\partial}{df(x_i)} \bar{\ell}(y_i, f(x_i))$$

Best Direction within $\mathcal{H}$

- Direct formulation:

$$h_t \in \underset{h \in \mathcal{H}}{\operatorname{argmin}} \frac{\sum_{i=1}^{n} \nabla_i L(y,f) h(x_i)}{\sqrt{\sum_{i=1}^{n} |h(x_i)|^2}} \left(= \frac{\langle \nabla L(y,f), h \rangle}{\|h\|} \right)$$

- Equivalent (least-squares) formulation: $h_t = -\beta_t h'_t$ with

$$(\beta_t, h'_t) \in \underset{(\beta,h) \in \mathbb{R} \times \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |\nabla_i L(y,f) - \beta h(x_i)|^2 \left(= \|\nabla L - \beta h\|^2 \right)$$

- Choice of the formulation will depend on $\mathcal{H}$...

Gradient Boosting of Classifiers

- **Assumptions:**
  - h is a binary classifier, $h(x) = \pm 1$ and thus $\|h\|^2 = n$.
  - $\bar{\ell}(y, f(x)) = l(yf(x))$ so that $\nabla_i L(y, f) = y_i l'(y_i f(x_i))$.
- Best direction h_t in $\mathcal{H}$ using the first formulation

$$h_t = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i \nabla_i L(y, f) h(x_i)$$

AdaBoost Type Minimization

- Best direction rewriting

$$\begin{aligned} h_t &= \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i l'(y_i f(x_i)) y_i h(x_i) \\ &= \underset{h \in \mathcal{H}}{\operatorname{argmin}} \sum_i (-l')(y_i f(x_i))(2\ell^{0/1}(y_i, h(x_i)) - 1) \end{aligned}$$

- **AdaBoost type weighted loss minimization** as soon as $(-l')(y_i f(x_i)) \geq 0$:

$$h_t = \operatorname{argmin} \sum_i (-l')(y_i f(x_i)) \ell^{0/1}(y_i, h(x_i))$$

Gradient Boosting

- **(Gradient) AdaBoost:** $\bar{\ell}(y, f) = \exp(-yf)$
  - $l(x) = \exp(-x)$ and thus $(-l')(y_i f(x_i)) = e^{-y_i f(x_i)} \geq 0$
  - h_t is the same as in AdaBoost
  - α_t also... (explicit computation)
- **LogitBoost:** $\bar{\ell}(y, f) = \log_2(1 + e^{-yf})$
  - $l(x) = \log_2(1 + e^{-x})$ and thus $(-l')(y_i f(x_i)) = \frac{e^{-y_i f(x_i)}}{\log(2)(1+e^{-y_i f(x_i)})} \geq 0$
  - Less weight on misclassified samples than in AdaBoost...
  - No explicit formula for α_t (line search)
  - Different path than with the (non-computable) classical boosting!
- **SoftBoost:** $\bar{\ell}(y, f) = \max(1 - yf, 0)$
  - $l(x) = \max(1 - x, 0)$ and $(-l')(y_i f(x_i)) = \mathbf{1}_{y_i f(x_i) \leq 1} \geq 0$
  - Do not use the samples that are sufficiently well classified!

Gradient Boosting and Least Squares

- Least squares formulation is often preferred when $|h| \neq 1$.

Least Squares Gradient Boosting

- Find $h_t = -\beta_t h'_t$ with

$$(\beta_t, h'_t) \in \underset{(\beta,h)\in\mathbb{R}\times\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |\nabla_i L(y, f) - \beta h(x_i)|^2$$

- Classical least squares if $\mathcal{H}$ is a finite dimensional vector space!
- Not a usual least squares in general but a classical regression problem!

- Numerical scheme depends on the loss...

Examples

- **Gradient L_2Boost:**
  - $\ell(y, f) = |y - f|^2$ and $\nabla_i L(y_i, f(x_i)) = -2(y_i - f(x_i))$:

$$(\beta_t, h'_t) \in \underset{(\beta,h)\in\mathbb{R}\times\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |2y_i - 2(f(x_i) - \beta/2h(x_i))|^2$$

  - $\alpha_t = -\beta_t/2$
  - Equivalent to classical L_2-Boosting
- **Gradient L_1Boost:**
  - $\ell(y, f) = |y - f|$ and $\nabla_i L(y_i, f(x_i)) = -\operatorname{sign}(y_i - f(x_i))$:

$$(\beta_t, h'_t) \in \underset{(\beta,h)\in\mathbb{R}\times\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |-\operatorname{sign}(y_i - f(x_i)) - \beta h(x_i)|^2$$

  - Robust to outliers...
- Classical choice for $\mathcal{H}$: Linear Model in which each h depends on a small subset of variables.

Gradient Boosting and Least Squares

- Least squares formulation can also be used in classification!
- Assumption:
  - $\ell(y, f(x)) = l(yf(x))$ so that $\nabla_i L(y_i, f(x_i)) = y_i l'(y_i f(x_i))$

Least Squares Gradient Boosting for Classifiers

- Least Squares formulation:

$$(\beta_t, h'_t) \in \underset{(\beta,h)\in\mathbb{R}\times\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |y_i l'(y_i f(x_i)) - \beta h(x_i)|^2$$

- Equivalent formulation:

$$(\beta_t, h'_t) \in \underset{(\beta,h)\in\mathbb{R}\times\mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} |(-l')(y_i f(x_i)) - (-\beta) y_i h(x_i))|^2$$

- **Intuition:** Modify misclassified examples without modifying too much the well-classified ones...

Boosting Variations

Stochastic Boosting

- **Idea:** change the learning set at each step.
- Two possible reasons:
  - Optimization over all examples too costly
  - Add variability to use an averaged solution
- Two different samplings:
  - Use sub-sampling, if you need to reduce the complexity
  - Use re-sampling, if you add variability...
- Stochastic Gradient name mainly used for the first case...

Second Order Boosting

- Replace the first order approximation by a second order one and avoid the line search...

XGBoost

- Very efficient boosting algorithm proposed by Chen and Guestrin in 2014.

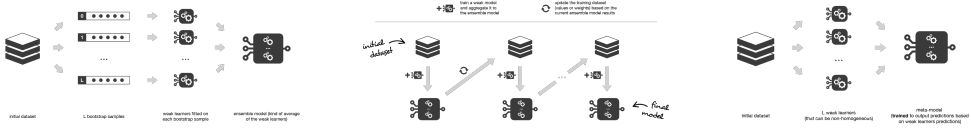
eXtreme Gradient Boosting

- Gradient boosting for a (penalized) smooth loss using a second order approximation and the least squares approximation.
- Reduced stepsize with a shrinkage of the *optimal* parameter.
- Feature subsampling.
- Weak learners:
  - Trees: limited depth, penalized size and parameters, fast approximate best split.
  - Linear model: elastic-net penalization.

- Excellent baseline!
- Lightgbm and CatBoost are also excellent similar choices!

Outline

Ensemble Methods



Ensemble Methods

- **Averaging:** combine several models by averaging (bagging, random forests,...)
- **Boosting:** construct a sequence of (weak) classifiers (XGBoost, LightGBM, CatBoost)
- **Stacking:** use the outputs of several models as features (tpot...)

- Loss of interpretability but gain in performance
- Beware of overfitting with stacking: the second learning step should be done with fresh data.
- No end to end optimization as in deep learning!

Source: J. Rocca

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

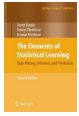
K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

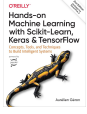
P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

References



T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

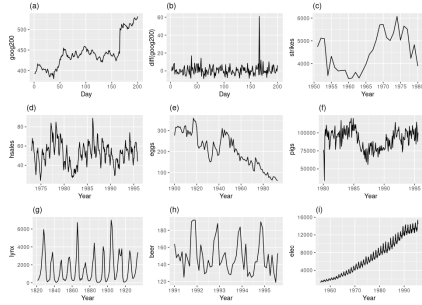
H. Zhang and B. Singer.
Recursive Partitioning and Applications.
Springer, 2010



A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (2nd ed.)
O'Reilly, 2019

Outline

Time Series

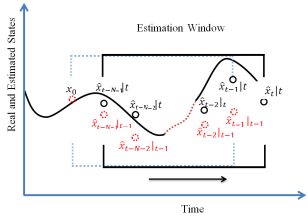


Time Series

- Sequence of values of the same entity across time.
- Values taken at regular interval, most of the time
- **Beware:** time dependency in the values!

Source: Hyndman and Athanasopoulos

Which Goals?



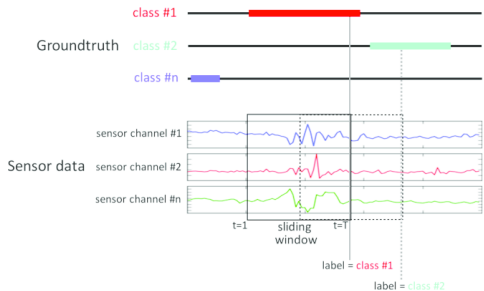
Goals

- Supervised:
  - Predict a value in the future,
  - Predict some values (a trajectory) in the future,
  - Predict a category in the future.
- Unsupervised:
  - Find break points,
  - Group some series together (possibly in real time)

- Using future values to act at a given time not allowed!

Source: N. Hashemian

Time Series and Structured Signals



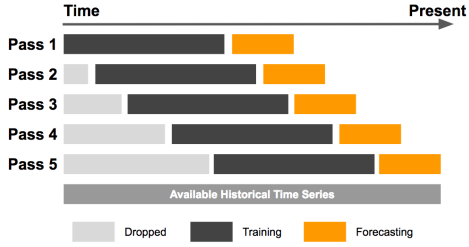
Structured Signals

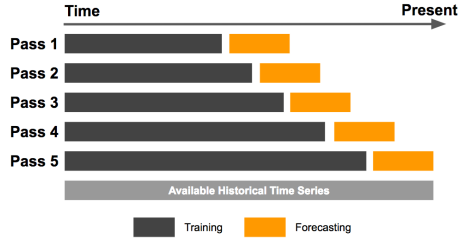
- Sequence of values of the same entity (spatially or temporaly).
- Decision can be taken a posteriori.
- No hard real-time constraints.

- Easier to deal with... but dependency with the data.

Source: D. Roggen

Time Series and Validation





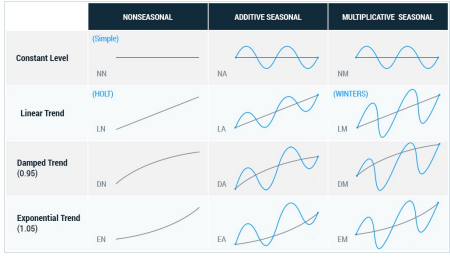
Cross Validation

- Never use the future... including for the validation.
- Classical Cross Validation is not working!
- Backtesting principle.

- Loss choice remains important.
- For structured data, safety buffer required between training and testing data.

Source: Uber

Trend and Seasonality



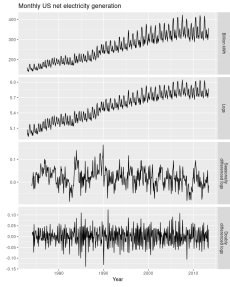
Trend and Seasonality

- Trend: long term evolution of average behavior.
- Seasonality: periodic variability around this mean.
- Residual: values after subtraction of the trend and the seasonality

- Need to estimate everything using only the past.

Source: E. Matveev

Stationarization



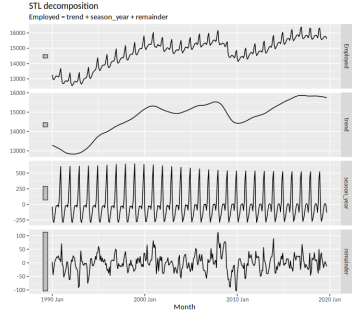
Stability in time assumption

- Required for learning...
- but not necessarily true.
- Often approximately correct after a transformation!

- Strongly data dependent!

Source: Hyndman and Athanasopoulos

Time Series Modeling



Models

- 3-layers approach: trend, seasonality and residuals.
- Decomposition not well specified...
- Several approaches for each layer!

Source: Hyndman and Athanasopoulos

Statistical Approach

$$\hat{X}_t \approx \sum_{j=1}^{p} \phi_j X_{t-j} + \sum_{k=1}^{q} \theta_k Z_{t-k} + \hat{Z}_t$$

Statistical Approach

- Most classical modeling.
- Combines past values of the sequence and a random noise.
- Explicit modeling of the variability!
- Complex estimation...

Source: H. Parra

Machine Learning Approach

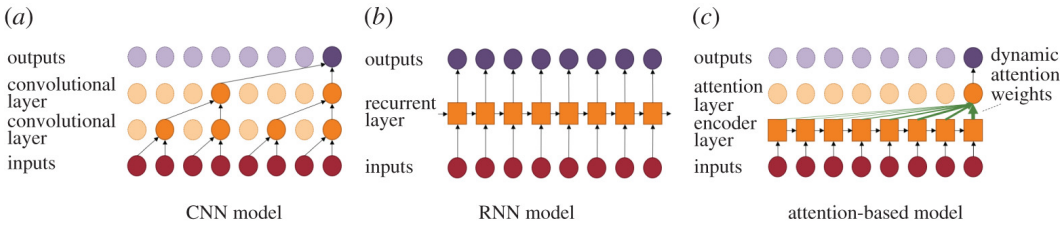
	Datetime	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	Count
0	2012-08-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8
1	2012-08-25 01:00:00	8.0	NaN	NaN	NaN	NaN	NaN	NaN	2
2	2012-08-25 02:00:00	2.0	8.0	NaN	NaN	NaN	NaN	NaN	6
3	2012-08-25 03:00:00	6.0	2.0	8.0	NaN	NaN	NaN	NaN	2
4	2012-08-25 04:00:00	2.0	6.0	2.0	8.0	NaN	NaN	NaN	2
5	2012-08-25 05:00:00	2.0	2.0	6.0	2.0	8.0	NaN	NaN	2
6	2012-08-25 06:00:00	2.0	2.0	2.0	6.0	2.0	8.0	NaN	2
7	2012-08-25 07:00:00	2.0	2.0	2.0	2.0	6.0	2.0	8.0	2
8	2012-08-25 08:00:00	2.0	2.0	2.0	2.0	2.0	6.0	2.0	6
9	2012-08-25 09:00:00	6.0	2.0	2.0	2.0	2.0	2.0	6.0	2

Machine Learning Approach

- Past taken into account only by feature engineering!
- Often using directly lagged values from the past.
- Variability not taken into account.
- Estimation with classical ML tools.

Source: A. Singh

Deep Learning Approach



Deep Learning Approach

- Past taken into account through the architecture.
- Explicit use of past values.
- Variability not taken into account.
- Huge choice for the architecture.

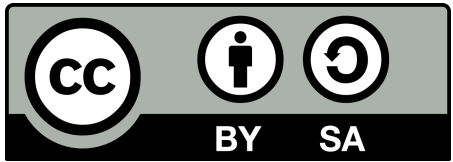
- Often trade-off performance/interpretability!

Source: B. Lim and S. Zohren

References

R. Hyndman and G. Athanopoulos.
Forecasting: principles and practice (3rd ed.)
OTexts, 2021

Licence and Contributors



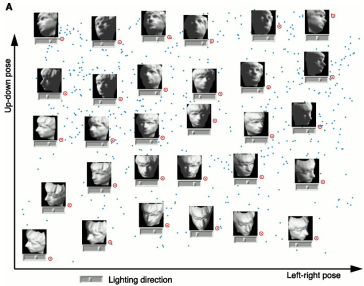
Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

Outline

Motivation



What is possible with data without labels?

- To group them?
- To visualize them in a 2 dimensional space?

Sources:: Unknown/Tenenbaum et al.

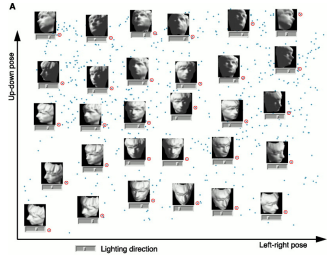
Marketing and Groups

To group them?

- **Data:** Base of customer data containing their properties and past buying records
- **Goal:** Use the customers *similarities* to find groups.
- **Clustering:** propose an explicit *grouping* of the customers
- **Visualization:** propose a representation of the customers so that the groups are *visible*. (Bonus)

Source: Unknown

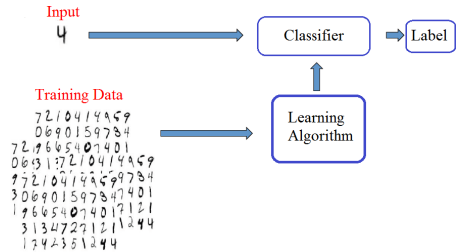
Image and Visualization



To visualize them?

- **Data:** Images of a single object
- **Goal:** Visualize the *similarities* between images.
- **Visualization:** propose a representation of the images so that similar images are *close*.
- **Clustering:** use this representation to cluster the images. (Bonus)

Source: Tenenbaum et al.



A definition by Tom Mitchell

A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

Source: Unknown

Supervised Learning

Experience, Task and Performance measure

- **Training data** : $\mathcal{D} = \{(\underline{X}_1, Y_1), \ldots, (\underline{X}_n, Y_n)\}$ (i.i.d. $\sim \mathbb{P}$)
- **Predictor**: $f : \mathcal{X} \to \mathcal{Y}$ measurable
- **Cost/Loss function**: $\ell(f(\underline{X}), Y)$ measure how well $f(\underline{X})$ *predicts* Y
- **Risk**:

$$\mathcal{R}(f) = \mathbb{E}[\ell(Y, f(\underline{X}))] = \mathbb{E}_X\Big[\mathbb{E}_{Y|\underline{X}}[\ell(Y, f(\underline{X}))]\Big]$$

- Often $\ell(f(\underline{X}), Y) = \|f(\underline{X}) - Y\|^2$ or $\ell(f(\underline{X}), Y) = \mathbf{1}_{Y \neq f(\underline{X})}$

Goal

- Learn a rule to construct a **predictor** $\widehat{f} \in \mathcal{F}$ from the training data $\mathcal{D}_n$ s.t. **the risk** $\mathcal{R}(\widehat{f})$ is **small on average** or with high probability with respect to $\mathcal{D}_n$.

Unsupervised Learning

Experience, Task and Performance measure

- **Training data** : $\mathcal{D} = \{\underline{X}_1, \ldots, \underline{X}_n\}$ (i.i.d. $\sim \mathbb{P}$)
- **Task**: ???
- **Performance measure**: ???

- No obvious task definition!

Tasks for this lecture

- **Dimension reduction:** construct a map of the data in a **low dimensional** space without **distorting** it too much.
- **Clustering (or unsupervised classification):** construct a **grouping** of the data in **homogeneous** classes.

Dimension Reduction

- **Training data** : $\mathcal{D} = \{\underline{X}_1, \dots, \underline{X}_n\} \in \mathcal{X}^n \quad (\text{i.i.d. } \sim \mathbb{P})$
- Space $\mathcal{X}$ of possibly high dimension.

Dimension Reduction Map

- Construct a map Φ from the space $\mathcal{X}$ into a space $\mathcal{X}'$ of **smaller dimension**:

$$\begin{aligned} \Phi : \quad & \mathcal{X} \to \mathcal{X}' \\ & \underline{X} \mapsto \Phi(\underline{X}) \end{aligned}$$

- Map can be defined only on the dataset.

Motivations

- Visualization of the data
- Dimension reduction (or embedding) before further processing

Dimension Reduction

- Need to control the **distortion** between $\mathcal{D}$ and $\Phi(\mathcal{D}) = \{\Phi(\underline{X}_1), \ldots, \Phi(\underline{X}_n)\}$

Distortion(s)

- Reconstruction error:
  - Construct $\widetilde{\Phi}$ from $\mathcal{X}'$ to $\mathcal{X}$
  - Control the error between $\underline{X}$ and its reconstruction $\widetilde{\Phi}(\Phi(\underline{X}))$
- Relationship preservation:
  - Compute a *relation* $\underline{X}_i$ and $\underline{X}_j$ and a *relation* between $\Phi(\underline{X}_i)$ and $\Phi(\underline{X}_j)$
  - Control the difference between those two *relations*.

- Lead to different constructions. . . .

Clustering

- **Training data** : $\mathcal{D} = \{\underline{X}_1, \ldots, \underline{X}_n\} \in \mathcal{X}^n \quad (\text{i.i.d. } \sim \mathbb{P})$
- Latent groups?

Clustering

- Construct a map f from $\mathcal{D}$ to $\{1, \ldots, K\}$ where K is a number of classes to be fixed:

$$f : \quad \underline{X}_i \mapsto k_i$$

- Similar to classification except:
  - no ground truth (no given labels)
  - label only elements of the dataset!

Motivations

- Interpretation of the groups
- Use of the groups in further processing

Clustering

- Need to define the **quality** of the cluster.
- No obvious measure!

Clustering quality

- Inner homogeneity: samples in the same group should be similar.
- Outer inhomogeneity: samples in two different groups should be different.

- Several possible definitions of similar and different.
- Often based on the distance between the samples.
- Example based on the Euclidean distance:
  - Inner homogeneity = intra-class variance,
  - Outer inhomogeneity = inter-class variance.
- **Beware:** choice of the number of clusters K often complex!

Bonus Task: Representation Learning

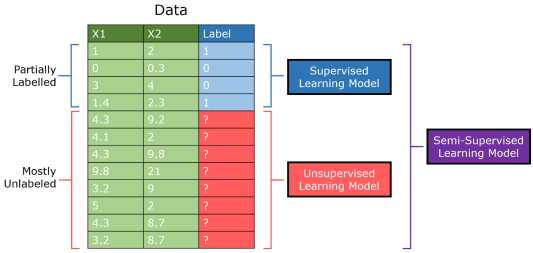
- **General observation:** most data do not have a label !
- **Example:** The number of images on which someone has described the content of the image is a *tiny fraction* of the images online.
- Labeling is very expensive and time consuming
- A lot of information can be extracted from the structure of the data, before seeing any label.

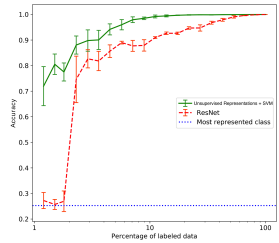
How can we leverage the large quantity of un-labeled data?

- Learn relevant features (= **representations**) in an unsupervised fashion
- Use those features to solve a supervised task with a fraction of labeled data.

- **Semi-supervised framework**
- $\looparrowright$ Very useful in practice, for images, time series, text.

Semi-supervised Framework





Semu-Supervised Framework

- With representation learned in an unsupervised fashion + a simple linear model, one can achieve the same performance with 10% of data labeled than with a fully annotated dataset.

- Complementary regularization based approaches also exist.

Source: Andre Ye, J.Y. Franceschi et al.

Unsupervised Learning is a Versatile Approach!

The learner is always right

- A subjective measure of performance
- Subjective choices for the algorithmic constraints (e.g., the type of transformation of the data we allow for low-dimensional representation, type of groups in clustering)
- $\Rightarrow$ Very difficult or impossible to tell which is the *best* method.

- Yet:
  - Extremely important in practice:
    - 90-99% of the data is un-labeled!
    - the tasks themselves are fundamental
  - Huge success in various fields (NLP, images...)

Unsupervised Learning is a Versatile Approach!

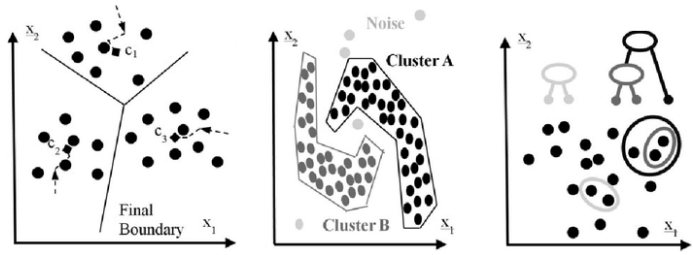
Lecture goals for the two main tasks

- Discussing possible choices of measures of performance and algorithmic constraints
- Understand the correspondences between those choices and a variety of classical algorithms
- For the simplest algorithms (PCA, k-means), get a precise mathematical understanding of the learning process.

Outline

Outline

What's a group?



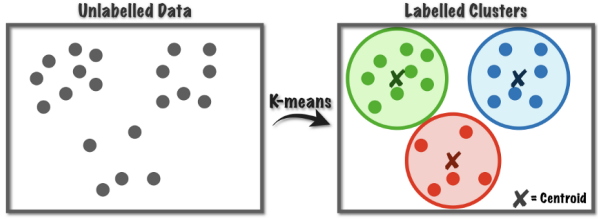
- No simple or unanimous definition!
- Require a notion of similarity/difference...

Three main approaches

- A group is a set of samples similar to a prototype.
- A group is a set of samples that can be linked by contiguity.
- A group can be obtained by fusing some smaller groups...

Source: J. Pitel

Prototype Approach



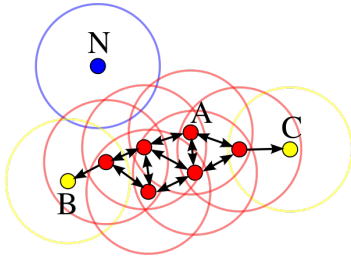
Prototype Approach

- A group is a set of samples similar to a prototype.
- Most classical instance: k-means algorithm.
- Principle: alternate prototype choice for the current groups and group update based on those prototypes.

- Number of groups fixed at the beginning
- No need to compare the samples between them!

Source: A. Jeffares

Contiguity Approach



Contiguity Approach

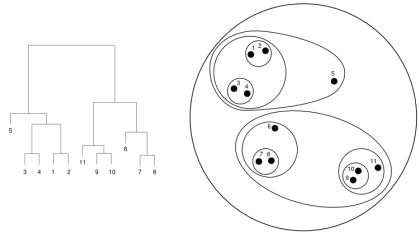
- A group is the set of samples that can be linked by contiguity.
- Most classical instance: DBScan
- Principle: group samples by contiguity if possible (proximity and density)

- Some samples may remain isolated.
- Number of groups controlled by the scale parameter.

Source: Wikipedia

DBSCAN: Density-Based Spatial Clustering of Applications with Noise

Agglomerative Approach

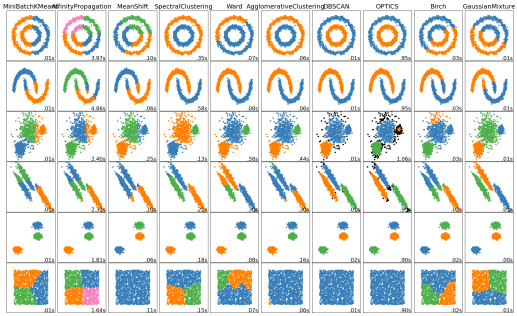


Agglomerative Approach

- A group can be obtained by fusing some smaller groups...
- Hierachical clustering principle: sequential merging of groups according to a *best merge* criterion

- Numerous variations on the merging criterion...
- Number of groups chosen afterward.

Choice of the method and of the number of groups



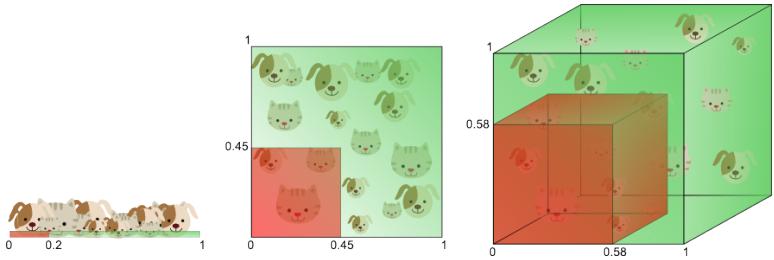
No method or number of groups is better than the others. . .

- Criterion not necessarily explicit!
- No cross validation possible
- Choice of the number of groups: a priori, heuristic, *based on the final usage. . .*

Source: Scikit-Learn

Outline

Dimensionality Curse



- **DISCLAIMER: Even if they are used everywhere beware of the usual distances in high dimension!**

Dimensionality Curse

- Previous approaches based on distances.
- Surprising behavior in high dimension: everything is ((often) as) far away.
- Beware of categories...

Source: Vision Dummy

Dimensionality Curse

- **DISCLAIMER: Even if they are used everywhere beware of the usual distances in high dimension!**

High Dimensional Geometry Curse

- Folks theorem: In high dimension, everyone is alone.
- Theorem: If $\underline{X}_1, \ldots, \underline{X}_n$ in the hypercube of dimension d such that their coordinates are i.i.d then

$$d^{-1/p}\left(\max \|\underline{X}_i - \underline{X}_j\|_p - \min \|\underline{X}_i - \underline{X}_j\|_p\right) = 0 + O_P\left(\sqrt{\frac{\log n}{d}}\right)$$

$$\frac{\min \|\underline{X}_i - \underline{X}_j\|_p}{\max \|\underline{X}_i - \underline{X}_j\|_p} = 1 + O_P\left(\sqrt{\frac{\log n}{d}}\right).$$

- When d is large, all the points are almost equidistant...
- Nearest neighbors are meaningless!

Outline

Visualization and Dimension Reduction

Visualization and Dimension Reduction

- How to view a dataset in high dimension !
- High dimension: dimension larger than 2!
- **Projection** onto a 2D space.

Source: F. Belardi

Visualization and Dimension Reduction

Visualization and Dimension Reduction

- How to view a dataset in high dimension !
- High dimension: dimension larger than 2!
- **Projection** onto a 2D space.

Source: F. Belardi

Visualization and Dimension Reduction

Visualization and Dimension Reduction

- How to view a dataset in high dimension !
- High dimension: dimension larger than 2!
- **Projection** onto a 2D space.

Source: F. Belardi

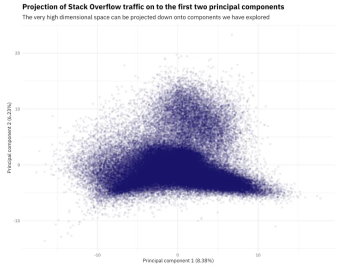
Visualization and Dimension Reduction

Visualization and Dimension Reduction

- How to view a dataset in high dimension !
- High dimension: dimension larger than 2!
- **Projection** onto a 2D space.

Source: F. Belardi

Principal Component Analysis



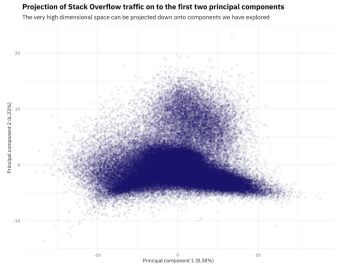
- Simple formula: $\tilde{X} = P(X - m)$

How to chose P?

- Maximising the dispersion of the points?
- Allowing to well reconstruct X from $\tilde{X}$?
- Preserving the relationship between the X through those between the $\tilde{X}$?

Source: J. Silge

Principal Component Analysis



- Simple formula: $\tilde{X} = P(X - m)$

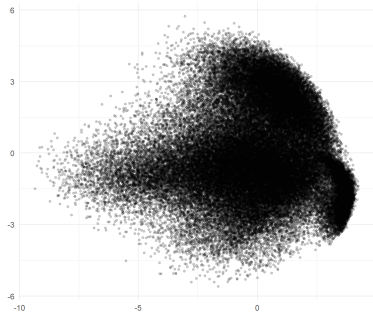
How to chose P?

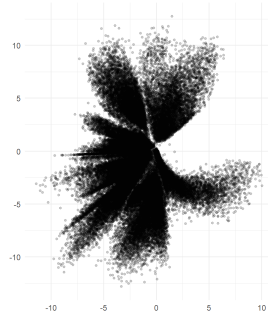
- Maximising the dispersion of the points?
- Allowing to well reconstruct X from $\tilde{X}$?
- Preserving the relationship between the X through those between the $\tilde{X}$?

- The 3 approaches yield the same solution!

Source: J. Silge

Reconstruction Approaches



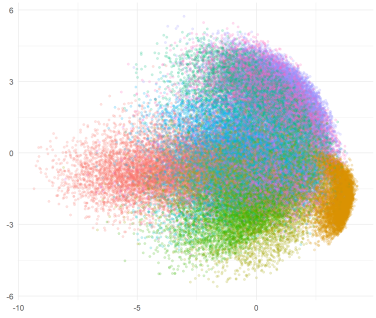


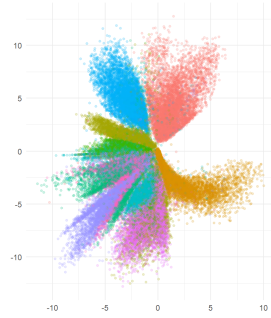
Reconstruction Approaches

- Learn a formula to encode and one formula to decode.
- Auto-encoder structure

- Yields a formula for new points.

Reconstruction Approaches



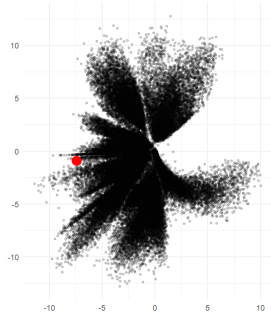


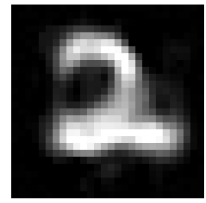
Reconstruction Approaches

- Learn a formula to encode and one formula to decode.
- Auto-encoder structure

- Yields a formula for new points.

Reconstruction Approaches



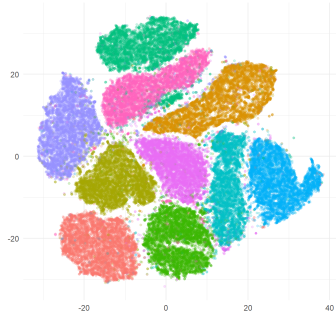


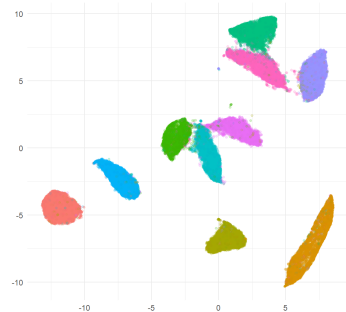
Reconstruction Approaches

- Learn a formula to encode and one formula to decode.
- Auto-encoder structure

- Yields a formula for new points.

Relationship Preservation Approaches



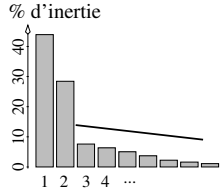


Relationship Preservation Approaches

- Based on the definition of the relationship notion (in both worlds).
- Huge flexibility

- Not always yields a formula for new points.

Choices of Methods and Dimension



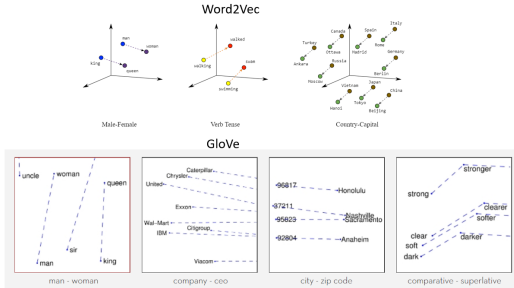
No Better Choice?

- Different criterion for different methods: impossible to use cross-validation.
- The larger the dimension the easier is to be faithful!
- In visualization, dimension 2 is the only choice.
- Heuristic criterion for the dimension choice: elbow criterion (no more gain), stability...

- Dimension Reduction is rarely used standalone but rather as a step in a predictive/prescriptive method.
- The dimension becomes an hyper-parameter of this method.

Source: E. Matzner-Løber

Representation Learning



Representation Learning

- How to transform arbitrary objects into numerical vectors?
- Objects: Categorical variables, Words, Images/Sounds...

- The two previous dimension reduction approaches can be used (given possibly a first simple high dimensional representation)

Source: R. Khandelwal

Outline

Dimension Reduction

- **Training data** : $\mathcal{D} = \{\underline{X}_1, \ldots, \underline{X}_n\} \in \mathcal{X}^n \quad (\text{i.i.d. } \sim \mathbb{P})$
- Space $\mathcal{X}$ of possibly high dimension.

Dimension Reduction Map

- Construct a map Φ from the space $\mathcal{X}$ into a space $\mathcal{X}'$ of **smaller dimension**:

$$\begin{aligned} \Phi : \quad & \mathcal{X} \to \mathcal{X}' \\ & \underline{X} \mapsto \Phi(\underline{X}) \end{aligned}$$

Criterion

- Reconstruction error
- Relationship preservation

Outline

How to Simplify?

A Projection Based Approach

- Observations: $\underline{X}_1, \ldots, \underline{X}_n \in \mathbf{R}^d$
- Simplified version: $\Phi(\underline{X}_1), \ldots, \Phi(\underline{X}_n) \in \mathbf{R}^d$ with Φ an affine projection preserving the mean $\Phi(\underline{X}) = P(\underline{X} - m) + m$ with $P^\top = P = P^2$ and $m = \frac{1}{n}\sum_i \underline{X}_i$.

How to choose P?

- **Inertia criterion**:
$$\max_P \sum_{i,j} \|\Phi(\underline{X}_i) - \Phi(\underline{X}_j)\|^2 ?$$
- **Reconstruction criterion**:
$$\min_P \sum_i \|\underline{X}_i - \Phi(\underline{X}_i)\|^2 ?$$
- **Relationship criterion**:
$$\min_P \sum_{i,j} |(\underline{X}_i - m)^\top (\underline{X}_j - m) - (\Phi(\underline{X}_i) - m)^\top (\Phi(\underline{X}_j) - m)|^2 ?$$

- **Rk:** Best solution is $P = I$! Need to reduce the rank of the projection to $d' < d \ldots$

Inertia criterion

- **Heuristic:** a good representation is such that the projected points are far apart.

Two views on inertia

- Inertia:

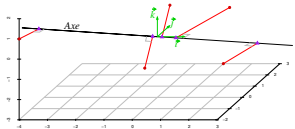
$$I = \frac{1}{2n^2} \sum_{i,j} \|\underline{X}_i - \underline{X}_j\|^2 = \frac{1}{n} \sum_{i=1}^{n} \|\underline{X}_i - m\|^2$$

- 2 times the mean squared distance to the mean = Mean squared distance between individual

Inertia criterion (Principal Component Analysis)

- Criterion: $\max_P \sum_{i,j} \frac{1}{2n^2} \|P\underline{X}_i - P\underline{X}_j\|^2 = \max_P \frac{1}{n} \sum_i \|P\underline{X}_i - m\|^2$
- **Solution:** Choose P as a projection matrix on the space spanned by the d' first eigenvectors of $\Sigma = \frac{1}{n} \sum_i (\underline{X}_i - m)(\underline{X}_i - m)^\top$

First Component of the PCA



- $\widetilde{\underline{X}} = m + a^\top(\underline{X} - m)a$ with $\|a\| = 1$
- Inertia: $\frac{1}{n}\sum_{i=1}^{n} a^\top(\underline{X}_i - m)(\underline{X}_i - m)^\top a$

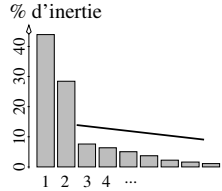
Principal Component Analysis: optimization of the projection

- Maximization of $\widetilde{I} = \frac{1}{n}\sum_{i=1}^{n} a^\top(\underline{X}_i - m)(\underline{X}_i - m)^\top a = a^\top \Sigma a$ with

  $\Sigma = \frac{1}{n}\sum_{i=1}^{n}(\underline{X}_i - m)(\underline{X}_i - m)^\top$ the empirical covariance matrix.
- Explicit optimal choice given by the eigenvector of the largest eigenvalue of Σ.

Source: E. Matzner-Løber

PCA



Principal Component Analysis : sequential optimization of the projection

- Explicit optimal solution obtain by the projection on the eigenvectors of the largest eigenvalues of Σ.
- Projected inertia given by the sum of those eigenvalues.

- Often fast decay of the eigenvalues: some dimensions are much more important than others.
- Not exactly the curse of dimensionality setting...
- Yet a lot of *small* dimension can drive the distance!

Reconstruction Criterion

- **Heuristic:** a good representation is such that the projected points are close to the original ones.

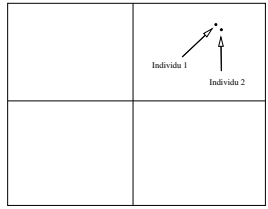
Reconstruction Criterion

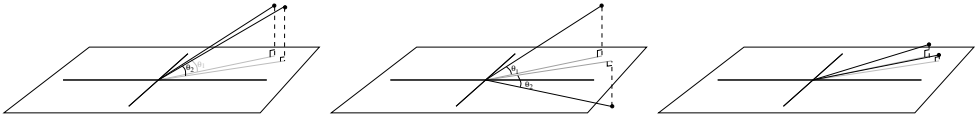
- Criterion: $\min_P \sum_i \frac{1}{n} \|\underline{X}_i - (P(\underline{X}_i - m) + m)\|^2 = \min_P \frac{1}{n} \sum_i \|(I - P)(\underline{X}_i - m)\|^2$
- **Solution:** Choose P as a projection matrix on the space spanned by the d' first eigenvectors of $\Sigma = \frac{1}{n} \sum_i (\underline{X}_i - m)(\underline{X}_i - m)^\top$

- Same solution with a different heuristic!
- Proof (Pythagora):

$$\sum_i \|\underline{X}_i - m\|^2 = \sum_i \left(\|P(\underline{X}_i - m)\|^2 + \|(I - P)(\underline{X}_i - m)\|^2 \right)$$

PCA, Reconstruction and Distances





Close projection doesn't mean close individuals!

- Same projections but different situations.
- Quality of the reconstruction measured by the angle with the projection space!

Source: E. Matzner-Løber

Relationship Criterion

- **Heuristic:** a good representation is such that the projected points scalar products are similar to the original ones.

Relationship Criterion (Multi Dimensional Scaling)

- Criterion: $\min_P \sum_{i,j} |(\underline{X}_i - m)^\top (\underline{X}_j - m) - (\Phi(\underline{X}_i) - m)^\top (\Phi(\underline{X}_j) - m)|^2$
- **Solution:** Choose P as a projection matrix on the space spanned by the d' first eigenvectors of $\Sigma = \frac{1}{n} \sum_i (\underline{X}_i - m)(\underline{X}_i - m)^\top$

- Same solution with a different heuristic!
- Much more involved justification!

Link with SVD

- PCA model: $\underline{X} - m \simeq P(\underline{X} - m)$
- **Prop:** $P = VV^\top$ with V an orthormal family in dimension d of size d'.
- PCA model with V: $\underline{X} - m \simeq VV^\top(\underline{X} - m)$ where $\tilde{\underline{X}} = V^\top(\underline{X} - m) \in \mathbb{R}^{d'}$
- Row vector rewriting: $\underline{X}^\top - m^\top \simeq \tilde{\underline{X}}^\top V^\top$

Matrix Rewriting and Low Rank Factorization

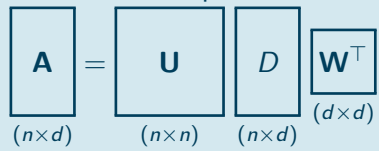
- Matrix rewriting

$$\begin{bmatrix} \underline{X}_1^\top - m^\top \\ \vdots \\ \underline{X}_n^\top - m^\top \end{bmatrix}_{(n\times d)} \simeq \begin{bmatrix} \tilde{\underline{X}}_1^\top \\ \vdots \\ \tilde{\underline{X}}_n^\top \end{bmatrix}_{(n\times d')} \boxed{\mathbf{V}^\top}_{(d'\times d)}$$

- Low rank matrix factorization! (Truncated SVD solution...)

SVD Decomposition

- Any matrix $n \times d$ matrix A can be decomposed as



with U and W two orthonormal matrices and D a *diagonal* matrix with decreasing values.

Low Rank Approximation

- The best low rank approximation or rank r is obtained by restriction of the matrices to the first r dimensions:

$$\underbrace{\boxed{\mathbf{A}}}_{(n\times d)} \simeq \underbrace{\boxed{\mathbf{U_r}}}_{(n\times r)} \underbrace{\boxed{D_{r,r}}}_{(r\times r)} \underbrace{\boxed{\mathbf{W_r}^\top}}_{(r\times d)}$$

  for both the operator norm and the Frobenius norm!

- PCA: Low rank approximation with Frobenius norm, $d' = r$ and

$$\begin{pmatrix} \underline{X}_1^\top - m^\top \\ \vdots \\ \underline{X}_n^\top - m^\top \end{pmatrix} \leftrightarrow A, \quad \begin{pmatrix} \underline{\tilde{X}}_1^\top \\ \vdots \\ \underline{\tilde{X}}_n^\top \end{pmatrix} \leftrightarrow \mathbf{U_r} D_{r,r}, \quad \mathbf{V}^\top \leftrightarrow \mathbf{W_r}^\top$$

SVD Decompositions

- Recentered data:
$$\mathbf{R} = \begin{pmatrix} \underline{X}_1^\top - m^\top \\ \vdots \\ \underline{X}_n^\top - m^\top \end{pmatrix} = UDW^\top$$
- Covariance matrix:
$$\Sigma = \mathbf{R}^\top \mathbf{R} = WD^\top DW$$
with $D^\top D$ diagonal.
- Gram matrix (matrix of scalar products):
$$G = \mathbf{R}\mathbf{R}^\top = UDD^\top U$$
with $DD^\top$ diagonal.

- Those are the same U, W and D, hence the link between all the approaches.

Outline

Reconstruction Error Approach

Goal

- Construct a map Φ from the space $\mathcal{X}$ into a space $\mathcal{X}'$ of **smaller dimension**:

$$\begin{aligned} \Phi : \quad & \mathcal{X} \to \mathcal{X}' \\ & \underline{X} \mapsto \Phi(\underline{X}) \end{aligned}$$

- Construct $\widetilde{\Phi}$ from $\mathcal{X}'$ to $\mathcal{X}$
- Control the error between $\underline{X}$ and its reconstruction $\widetilde{\Phi}(\Phi(\underline{X}))$

- Canonical example for $\underline{X} \in \mathbb{R}^d$: find Φ and $\widetilde{\Phi}$ in a parametric family that minimize

$$\frac{1}{n} \sum_{i=1}^{n} \|\underline{X}_i - \widetilde{\Phi}(\Phi(\underline{X}_i))\|^2$$

Principal Component Analysis

- $\mathcal{X} \in \mathbb{R}^d$ and $\mathcal{X}' = \mathbb{R}^{d'}$
- Affine model $\underline{X} \sim m + \sum_{l=1}^{d'} \underline{X}'^{(l)} V^{(l)}$ with $(V^{(l)})$ an orthonormal family.
- Equivalent to:
$$\Phi(\underline{X}) = V^\top(\underline{X} - m) \quad \text{and} \quad \widetilde{\Phi}(\underline{X}') = m + V\underline{X}'$$
- Reconstruction error criterion:
$$\frac{1}{n}\sum_{i=1}^{n} \|\underline{X}_i - (m + VV^\top(\underline{X}_i - m)\|^2$$
- **Explicit solution:** m is the empirical mean and V is any orthonormal basis of the space spanned by the d' first eigenvectors (the one with largest eigenvalues) of the empirical covariance matrix $\frac{1}{n}\sum_{i=1}^{n}(\underline{X}_i - m)(\underline{X}_i - m)^\top$.

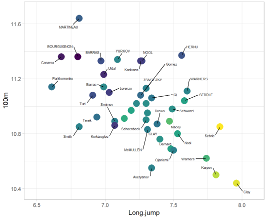
Principal Component Analysis

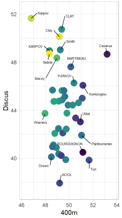
PCA Algorithm

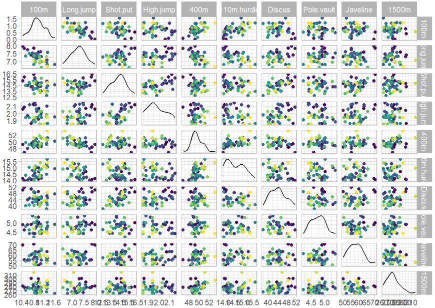
- Compute the empirical mean $m = \frac{1}{n}\sum_{i=1}^{n} \underline{X}_i$
- Compute the empirical covariance matrix $\frac{1}{n}\sum_{i=1}^{n}(\underline{X}_i - m)(\underline{X}_i - m)^\top$.
- Compute the d' first eigenvectors of this matrix: $V^{(1)}, \ldots, V^{(d')}$
- Set $\Phi(\underline{X}) = V^\top(\underline{X} - m)$

- Complexity: $O(n(d + d^2) + d'd^2)$
- Interpretation:
  - $\Phi(\underline{X}) = V^\top(\underline{X} - m)$: coordinates in the restricted space.
  - $V^{(i)}$: influence of each original coordinates in the ith new one.
- **Scaling:** This method is not invariant to a scaling of the variables! It is custom to normalize the variables (at least within groups) before applying PCA.

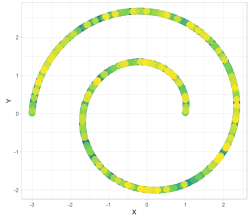
Decathlon

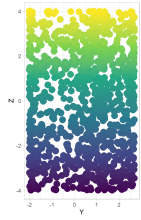

100m
11.6
11.2
10.8
10.4
6.5
7.0
7.5
8.0
Long.jump
MARTINEAU
BOURGUIGNON
Casarsa
BARRAS
YURKOV
NOOL
HERNU
Parkhomenko
WARNERS
SEBRLE
Karpov
Clay

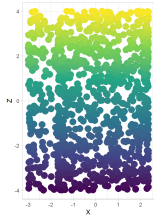

Discus
52
48
44
40
46
48
50
52
54
400m
Karpov
CLAY
Casarsa
MARTINEAU
YURKOV
BOURGUIGNON
Parkhomenko
NOOL

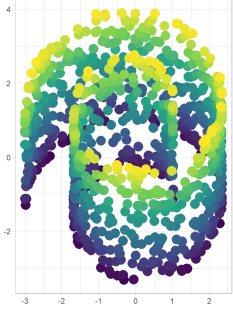

100m
Long.jump
Shot.put
High.jump
400m
Discus
Pole.vault
Javeline
1500m

Swiss Roll

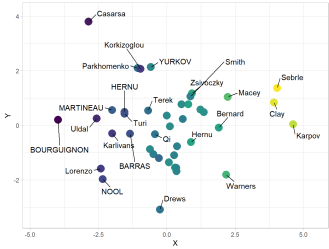




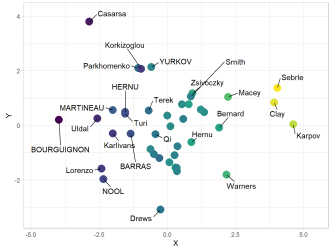




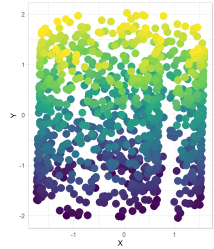
Principal Component Analysis



Decathlon



Decathlon
Renormalized



Swiss Roll

Multiple Factor Analysis

- PCA assumes $\mathcal{X} = \mathbb{R}^d$!
- How to deal with categorical values?
- MFA = PCA with clever coding strategy for categorical values.

Categorical value code for a single variable

- Classical redundant dummy coding:
$$\underline{X} \in \{1, \dots, V\} \mapsto P(\underline{X}) = (\mathbf{1}_{\underline{X}=1}, \dots, \mathbf{1}_{\underline{X}=V})^\top$$
- Compute the mean (i.e. the empirical proportions): $\overline{P} = \frac{1}{n} \sum_{i=1}^n P(\underline{X}_i)$
- *Renormalize* $P(\underline{X})$ by $1/\sqrt{(V-1)\overline{P}}$:
$$P(\underline{X}) = (\mathbf{1}_{\underline{X}=1}, \dots \mathbf{1}_{\underline{X}=V}) \mapsto \left(\frac{\mathbf{1}_{\underline{X}=1}}{\sqrt{(V-1)\overline{P}_1}}, \dots, \frac{\mathbf{1}_{\underline{X}=V}}{\sqrt{(V-1)\overline{P}_V}} = P^r(\underline{X}) \right)$$

- χ^2 type distance!

Multiple Factor Analysis

- PCA becomes the minimization of

$$\frac{1}{n}\sum_{i=1}^{n}\|P^r(\underline{X}_i) - (m + VV^\top(P^r(\underline{X}_i) - m))\|^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{v=1}^{V}\frac{\left|\mathbf{1}_{\underline{X}_i=v} - (m' + \sum_{l=1}^{d'} V^{(l)\top}(P(\underline{X}_i) - m')V^{(l,v)})\right|^2}{(V-1)\overline{P}_v}$$

- Interpretation:
  - $m' = \overline{P}$
  - $\Phi(\underline{X}) = V^\top(P^r(\underline{X}) - m)$: coordinates in the restricted space.
  - $V^{(l)}$ can be interpreted s as a probability profile.
- Complexity: $O(n(V + V^2) + d'V^2)$
- Link with Correspondence Analysis (CA)

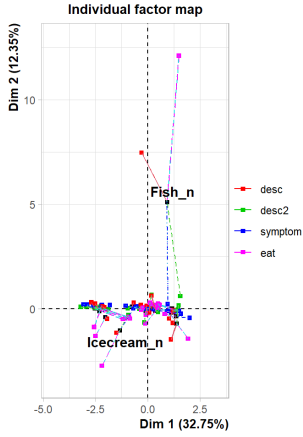
Multiple Factor Analysis

MFA Algorithm

- Redundant dummy coding of each categorical variable.
- Renormalization of each block of dummy variable.
- Classical PCA algorithm on the resulting variables

- Interpretation as a reconstruction error with a rescaled/χ^2 metric.
- Interpretation:
  - $\Phi(\underline{X}) = V^\top(P^r(\underline{X}) - m)$: coordinates in the restricted space.
  - $V^{(l)}$: influence of each modality/variable in the ith new coordinates.
- **Scaling:** This method is not invariant to a scaling of the continuous variables! It is custom to normalize the variables (at least within groups) before applying PCA.

Multiple Factor Analysis



PCA Model

- PCA: Linear model assumption

$$\underline{X} \simeq m + \sum_{l=1}^{d'} \underline{X}^{\prime,(l)} V^{(l)} = m + V\underline{X}'$$

- with
  - $V^{(l)}$ orthonormal
  - $\underline{X}^{\prime,(l)}$ without constraints.

- Two directions of extension:
  - Other constraints on V (or the coordinates in the restricted space): ICA, NMF, Dictionary approach
  - PCA on a non-linear image of $\underline{X}$: kernel-PCA
- Much more complex algorithm!

Non Linear PCA

ICA (Independent Component Analysis)

- Linear model assumption

$$\underline{X} \simeq m + \sum_{l=1}^{d'} \underline{X}'^{,(l)} V^{(l)} = m + V\underline{X}'$$

- with
  - $V^{(l)}$ without constraints.
  - $\underline{X}'^{,(l)}$ *independent*

NMF (Non Negative Matrix Factorization)

- (Linear) Model assumption

$$\underline{X} \simeq \sum_{l=1}^{d'} \underline{X}'^{,(l)} V^{(l)} = V\underline{X}'$$

- with
  - $V^{(l)}$ non-negative
  - $\underline{X}'^{,(l)}$ non-negative.

Non Linear PCA

Dictionary

- (Linear) Model assumption

$$\underline{X} \simeq m + \sum_{l=1}^{d'} \underline{X}'^{,(l)} V^{(l)} = m + V\underline{X}'$$

- with
  - $V^{(l)}$ without constraints
  - $\underline{X}'$ sparse (with a lot of 0)

kernel PCA

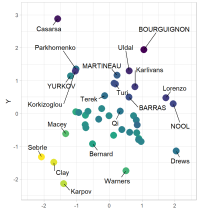
- Linear model assumption

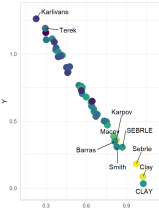
$$\Psi(\underline{X} - m) \simeq \sum_{l=1}^{d'} \underline{X}'^{,(l)} V^{(l)} = V\underline{X}'$$

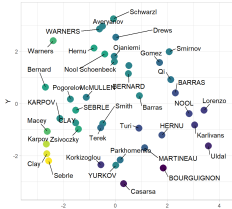
- with
  - $V^{(l)}$ orthonormal
  - $\underline{X}'_l$ without constraints.

Non Linear PCA

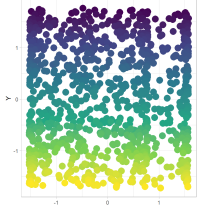
Decathlon

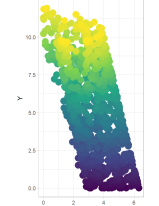


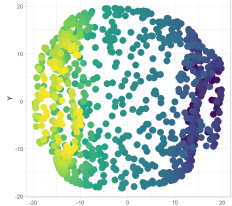




Swiss Roll







ICA NMF Kernel PCA

Deep Auto Encoder

- Construct a map Φ with a **NN** from the space $\mathcal{X}$ into a space $\mathcal{X}'$ of smaller dimension:

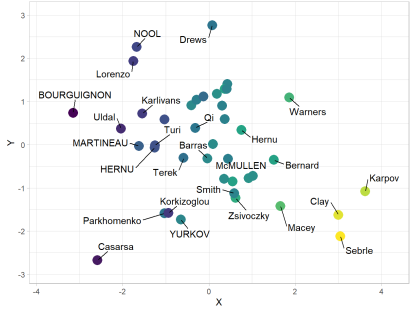
$$\begin{aligned} \Phi: \quad & \mathcal{X} \to \mathcal{X}' \\ & \underline{X} \mapsto \Phi(\underline{X}) \end{aligned}$$

- Construct $\widetilde{\Phi}$ with a **NN** from $\mathcal{X}'$ to $\mathcal{X}$
- Control the error between $\underline{X}$ and its reconstruction $\widetilde{\Phi}(\Phi(\underline{X}))$:

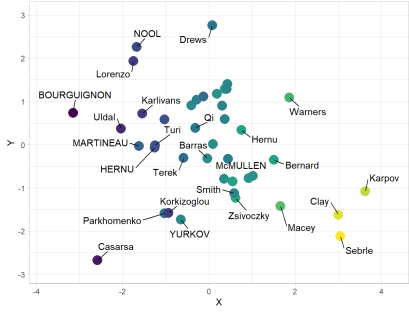
$$\frac{1}{n}\sum_{i=1}^{n} \|\underline{X}_i - \widetilde{\Phi}(\Phi(\underline{X}_i))\|^2$$

- Optimization by gradient descent.
- NN can be replaced by another parametric function...

Deep Auto Encoder



Shallow Auto Encoder (PCA)



Deep Auto Encoder

Outline

Pairwise Relation

- Different point of view!
- Focus on pairwise relation $\mathcal{R}(\underline{X}_i, \underline{X}_j)$.

Distance Preservation

- Construct a map Φ from the space $\mathcal{X}$ into a space $\mathcal{X}'$ of **smaller dimension**:

$$\begin{aligned} \Phi : \quad \mathcal{X} &\to \mathcal{X}' \\ \underline{X} &\mapsto \Phi(\underline{X}) = \underline{X}' \end{aligned}$$

- such that

$$\mathcal{R}(\underline{X}_i, \underline{X}_j) \sim \mathcal{R}'(\underline{X}'_i, \underline{X}'_j)$$

- Most classical version (MDS):
  - Scalar product relation: $\mathcal{R}(\underline{X}_i, \underline{X}_j) = (\underline{X}_i - m)^\top (\underline{X}_j - m)$
  - Linear mapping $\underline{X}' = \Phi(\underline{X}) = V^\top (\underline{X} - m)$.
  - Euclidean scalar product matching:

$$\frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left| (\underline{X}_i - m)^\top (\underline{X}_j - m) - (\underline{X}'_i)^\top \underline{X}'_j \right|^2$$

- Φ often defined only on $\mathcal{D}\ldots$

MultiDimensional Scaling

MDS Heuristic

- Match the *scalar* products:
$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left|(\underline{X}_i - m)^\top(\underline{X}_j - m) - \underline{X}_i{}'^\top \underline{X}'_j\right|^2$$
- Linear method: $\underline{X}' = U^\top(\underline{X} - m)$ with U orthonormal

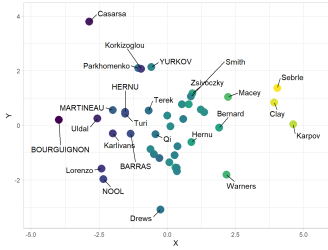
- **Beware:** $\underline{X}$ can be unknown, only the scalar products are required!
- Resulting criterion: minimization in $U^\top(\underline{X}_i - m)$ of
$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left|(\underline{X}_i - m)^\top(\underline{X}_j - m) - (\underline{X}_i - m)^\top UU^\top(\underline{X}_j - m)\right|^2$$
without using explicitly $\underline{X}$ in the algorithm...
- Explicit solution obtained through the eigendecomposition of the know Gram matrix $(\underline{X}_i - m)^\top(\underline{X}_j - m)$ by keeping only the d' largest eigenvalues.

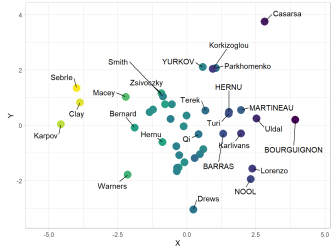
MultiDimensional Scaling

- In this case, MDS yields the same result as the PCA (but with different inputs, distance between observation vs correlations)!
- **Explanation:** Same SVD problem up to a transposition:
  - MDS
  $$\overline{X}_{(n)}{}^\top \overline{X}_{(n)} \sim \overline{X}_{(n)}{}^\top U U^\top \overline{X}_{(n)}$$
  - PCA
  $$\overline{X}_{(n)} \overline{X}_{(n)}{}^\top \sim U^\top \overline{X}_{(n)} \overline{X}_{(n)}{}^\top U$$
- Complexity: PCA $O((n+d')d^2)$ vs MDS $O((d+d')n^2)\ldots$

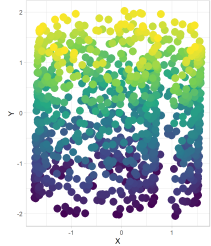
MultiDimensional Scaling

Decathlon

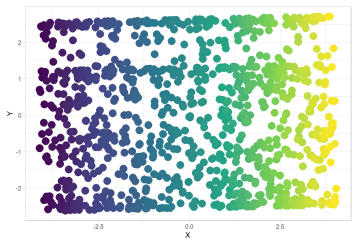




Swiss Roll



PCA



MDS

Generalized MDS

- Preserving the scalar products amounts to preserve the Euclidean distance.
- Easier **generalization** if we work in terms of distance!

Generalized MDS

- Generalized MDS:
  - Distance relation: $\mathcal{R}(\underline{X}_i, \underline{X}_j) = d(\underline{X}_i, \underline{X}_j)$
  - Linear mapping $\underline{X}' = \Phi(\underline{X}) = V^\top(\underline{X} - m)$.
  - Euclidean matching:

$$\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left|d(\underline{X}_i, \underline{X}_j) - d'(\underline{X}'_i, \underline{X}'_j)\right|^2$$

- Strong connection (but no equivalence) with MDS when $d(x, y) = \|x - y\|^2$!
- **Minimization:** Simple gradient descent can be used (can be stuck in local minima).

ISOMAP

- MDS: equivalent to PCA (but more expensive) if $d(x, y) = \|x - y\|^2$!
- ISOMAP: use a *localized* distance instead to limit the influence of very far point.

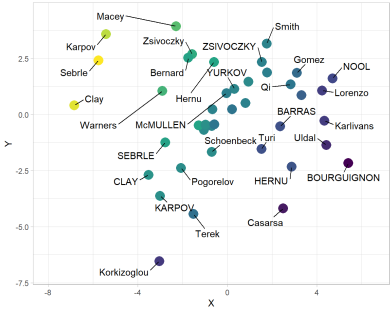
ISOMAP

- For each point $\underline{X}_i$, define a neighborhood $\mathcal{N}_i$ (either by a distance or a number of points) and let

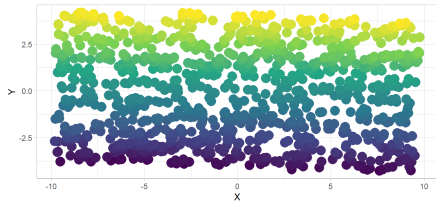
$$d_0(\underline{X}_i, \underline{X}_j) = \begin{cases} +\infty & \text{if } \underline{X}_j \notin \mathcal{N}_i \\ \|\underline{X}_i - \underline{X}_j\|^2 & \text{otherwise} \end{cases}$$

- Compute the shortest path distance for each pair.
- Use the MDS algorithm with this distance

ISOMAP



Decathlon



Swiss Roll

Random Projection

Random Projection Heuristic

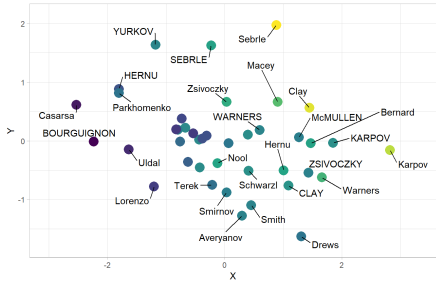
- Draw at random d' unit vector (direction) U_i.
- Use $\underline{X}' = U^\top(\underline{X} - m)$ with $m = \frac{1}{n}\sum_{i=1}^n \underline{X}_i$

- **Property:** If $\underline{X}$ lives in a space of dimension d'', then, as soon as, $d' \sim d'' \log(d'')$,

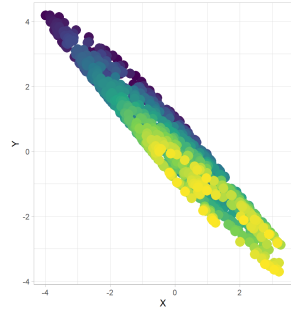
$$\|\underline{X}_i - \underline{X}_j\|^2 \sim \frac{d}{d'}\|\underline{X}'_i - \underline{X}'_j\|^2$$

- Do not really use the data!

Random Projection



Decathlon



Swiss Roll

t-Stochastic Neighbor Embedding

SNE heuristic

- From $\underline{X}_i \in \mathcal{X}$, construct a set of conditional probability:

$$P_{j|i} = \frac{e^{-\|\underline{X}_i - \underline{X}_j\|^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\underline{X}_i - \underline{X}_k\|^2/2\sigma_i^2}} \qquad P_{i|i} = 0$$

- Find $\underline{X}'_i$ in $\mathbb{R}^{d'}$ such that the set of conditional probability:

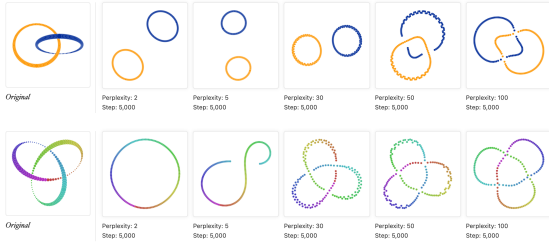
$$Q_{j|i} = \frac{e^{-\|\underline{X}'_i - \underline{X}'_j\|^2/2\sigma_i^2}}{\sum_{k \neq i} e^{-\|\underline{X}'_i - \underline{X}'_k\|^2/2\sigma_i^2}} \qquad Q_{i|i} = 0$$

  is close from P.

- **t-SNE:** use a Student-t term $(1 + \|\underline{X}'_i - \underline{X}'_j\|^2)^{-1}$ for $\underline{X}'_i$
- Minimize the Kullback-Leibler divergence $(\sum_{i,j} P_{j|i} \log \frac{P_{j|i}}{Q_{j|i}})$ by a simple gradient descent (can be stuck in local minima).
- Parameters σ_i such that $H(P_i) = -\sum_{j=1}^{n} P_{j|i} \log P_{j|i} = \text{cst}$.

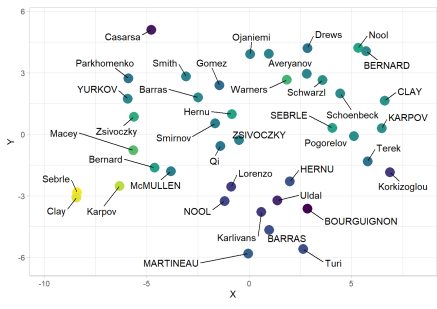
t-Stochastic Neighbor Embedding

- Very successful/ powerful technique in practice
- Convergence may be long, unstable, or strongly depending on parameters.
- See this distill post for many impressive examples

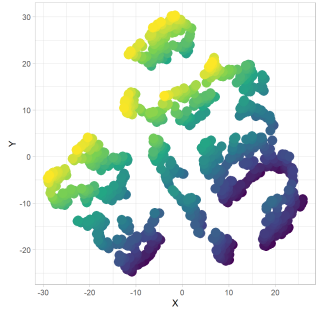


Representation depending on t-SNE parameters

t-SNE



Decathlon



Swiss Roll

UMAP

- Topological Data Analysis inspired.

Uniform Manifold Approximation and Projection

- Define a notion of asymmetric scaled local proximity between neighbors:
  - Compute the k-neighborhood of $\underline{X}_i$, its diameter σ_i and the distance ρ_i between $\underline{X}_i$ and its nearest neighbor.
  - Define

$$w_i(\underline{X}_i, \underline{X}_j) = \begin{cases} e^{-(d(\underline{X}_i, \underline{X}_j) - \rho_i)/\sigma_i} & \text{for } \underline{X}_j \text{ in the } k\text{-neighborhood} \\ 0 & \text{otherwise} \end{cases}$$

- Symmetrize into a *fuzzy* nearest neighbor criterion

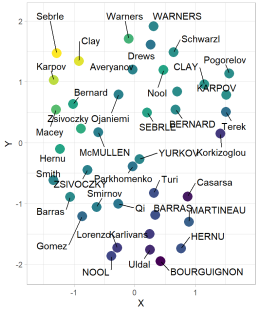
$$w(\underline{X}_i, \underline{X}_j) = w_i(\underline{X}_i, \underline{X}_j) + w_j(\underline{X}_j, \underline{X}_i) - w_i(\underline{X}_i, \underline{X}_j) w_j(\underline{X}_j, \underline{X}_i)$$

- Determine the points $\underline{X}'_i$ in a low dimensional space such that

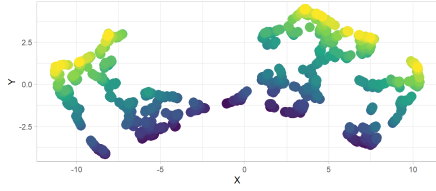
$$\sum_{i \neq j} w(\underline{X}_i, \underline{X}_j) \log \left(\frac{w(\underline{X}_i, \underline{X}_j)}{w'(\underline{X}'_i, \underline{X}'_j)} \right) + (1 - w(\underline{X}_i, \underline{X}_j)) \log \left(\frac{(1 - w(\underline{X}_i, \underline{X}_j))}{(1 - w'(\underline{X}'_i, \underline{X}'_j))} \right)$$

- Can be performed by local gradient descent.

UMAP



Decathlon



Swiss Roll

Graph based

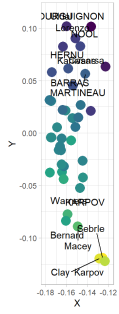
Graph heuristic

- Construct a graph with weighted edges $w_{i,j}$ measuring the *proximity* of $\underline{X}_i$ and $\underline{X}_j$ ($w_{i,j}$ large if close and 0 if there is no information).
- Find the points $\underline{X}'_i \in \mathbb{R}^{d'}$ minimizing

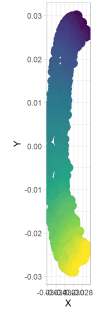
$$\frac{1}{n}\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{i,j}\|\underline{X}'_i - \underline{X}'_j\|^2$$

- Need of a constraint on the size of $\underline{X}'_i \ldots$
- Explicit solution through linear algebra: d' eigenvectors with smallest eigenvalues of the Laplacian of the graph $D - W$, where D is a diagonal matrix with $D_{i,i} = \sum_j w_{i,j}$.
- Variation on the definition of the Laplacian...

Graph



Decathlon



Swiss Roll

Outline

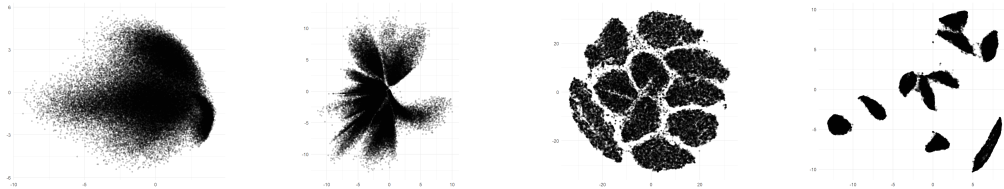
How to Compare Different Dimensionality Reduction Methods ?

- **Difficult!** Once again, the metric is very subjective.

However, a few possible attempts

- Did we preserve a lot of inertia with only a few directions?
- Do those directions *make sense* from an expert point of view?
- Do the low dimension representation *preserve* some important information?
- Are we better on **subsequent task**?

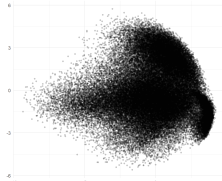
A Challenging Example: MNIST



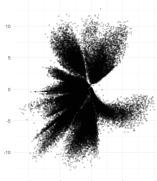
MNIST Dataset

- Images of 28×28 pixels.
- No label used!
- 4 different embeddings.

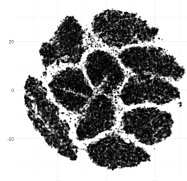
A Challenging Example: MNIST



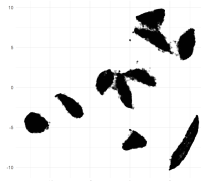
PCA



autoencoder



t-SNE

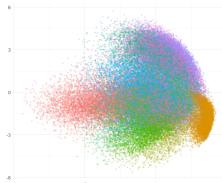


UMAP

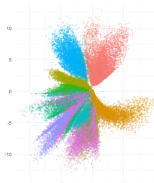
MNIST Dataset

- Images of 28×28 pixels.
- No label used!
- 4 different embeddings.

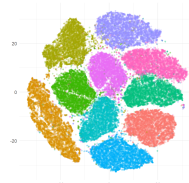
A Challenging Example: MNIST



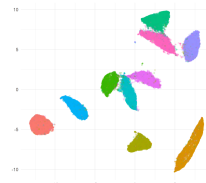
PCA



autoencoder



t-SNE



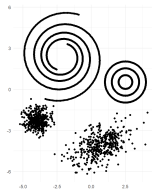
UMAP

MNIST Dataset

- Images of 28×28 pixels.
- No label used!
- 4 different embeddings.

- Quality evaluated by visualizing the true labels **not used to obtain the embeddings.**
- Only a few labels could have been used.

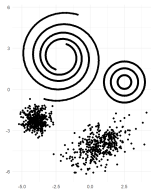
A Simpler Example: A 2D Set



Cluster Dataset

- Set of points in 2D.
- No label used!
- 3 different embeddings.

A Simpler Example: A 2D Set



PCA

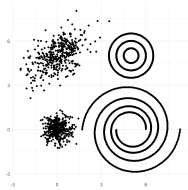
t-SNE

UMAP

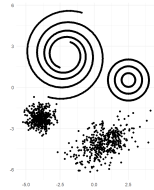
Cluster Dataset

- Set of points in 2D.
- No label used!
- 3 different embeddings.

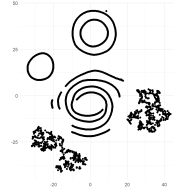
A Simpler Example: A 2D Set



Original



PCA



t-SNE

UMAP

Cluster Dataset

- Set of points in 2D.
- No label used!
- 3 different embeddings.

- Quality evaluated by stability...

Outline

Clustering

- **Training data** : $\mathcal{D} = \{\underline{X}_1, \ldots, \underline{X}_n\} \in \mathcal{X}^n \quad (\text{i.i.d. } \sim \mathbb{P})$
- Latent groups?

Clustering

- Construct a map f from $\mathcal{D}$ to $\{1, \ldots, K\}$ where K is a number of classes to be fixed:

$$f : \quad \underline{X}_i \mapsto k_i$$

Motivations

- Interpretation of the groups
- Use of the groups in further processing

- Several strategies possible!
- Can use dimension reduction as a preprocessing.

Outline

Partition Based

Partition Heuristic

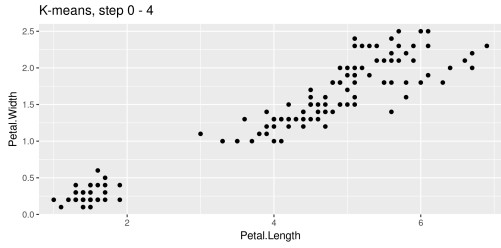
- Clustering is defined by a partition in K classes...
- that minimizes a homogeneity criterion.

K- Means

- Cluster k defined by a *center* μ_k.
- Each sample is associated to the closest center.
- Centers defined as the minimizer of $\sum_{i=1}^{n} \min_k \|\underline{X}_i - \mu_k\|^2$

- Iterative scheme (Loyd):
  - Start by a (pseudo) random choice for the centers μ_k
  - Assign each samples to its nearby center
  - Replace the center of a cluster by the mean of its assigned samples.
  - Repeat the last two steps until convergence.

Partition Based



Partition based

- Other schemes:
  - McQueen: modify the mean each time a sample is assigned to a new cluster.
  - Hartigan: modify the mean by removing the considered sample, assign it to the nearby center and recompute the new mean after assignment.

A good initialization is crucial!

- Initialize by samples.
- k-Mean++: try to take them as separated as possible.
- No guarantee to converge to a global optimum: repeat and keep the best result!

- Complexity : $O(n \times K \times T)$ where T is the number of steps in the algorithm.

Partition based

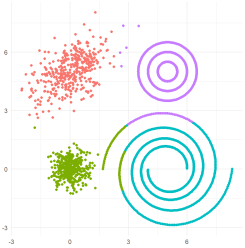
- k-Medoid: use a sample as a center
  - PAM: for a given cluster, use the sample that minimizes the intra distance (sum of the squared distance to the other points)
  - Approximate medoid: for a given cluster, assign the point that is the closest to the mean.

Complexity

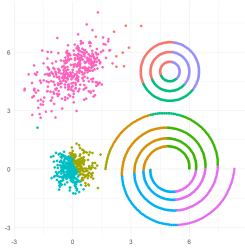
- PAM: $O(n^2 \times T)$ in the worst case!
- Approximate medoid: $O(n \times K \times T)$ where T is the number of steps in the algorithm.

- **Remark:** Any distance can be used... but the complexity of computing the centers can be very different.

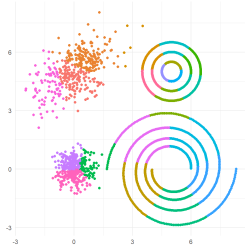
K-Means



$k = 4$



$k = 10$



$k = 10$

Model Based

Model Heuristic

- Use a generative model of the data:

$$\mathbb{P}(\underline{X}) = \sum_{k=1}^{K} \pi_k \mathbb{P}_{\theta_k}(\underline{X}|k)$$

  where π_k are proportions and $\mathbb{P}_\theta(\underline{X}|k)$ are parametric probability models.
- Estimate those parameters (often by a ML principle).
- Assign each observation to the class maximizing the a posteriori probability (obtained by Bayes formula)

$$\frac{\widehat{\pi_k}\mathbb{P}_{\widehat{\theta_k}}(\underline{X}|k)}{\sum_{k'=1}^{K}\widehat{\pi_{k'}}\mathbb{P}_{\widehat{\theta_{k'}}}(\underline{X}|k')}$$

- Link with Generative model in supervised classification!

Model Based

- Large choice of parametric models.

Gaussian Mixture Model

- Use
$$\mathbb{P}_{\theta_k}\left(\vec{\underline{X}}|k\right) \sim \mathcal{N}(\mu_k, \Sigma_k)$$
with $\mathcal{N}(\mu, \Sigma)$ the Gaussian law of mean μ and covariance matrix Σ.

- Efficient optimization algorithm available (EM)
- Often some constraint on the covariance matrices: identical, with a similar structure...
- Strong connection with K-means when the covariance matrices are assumed to be the same multiple of the identity.

Model Based

Probabilistic latent semantic analysis (PLSA)

- Documents described by their word counts w
- Model:

$$\mathbb{P}(w) = \sum_{k=1}^{K} \pi_k \mathbb{P}_{\theta_k}(w|k)$$

with k the (hidden) topic, π_k a topic probability and $\mathbb{P}_{\theta_k}(w|k)$ a multinomial law for a given topic.

- Clustering according to

$$\mathbb{P}(k|w) = \frac{\widehat{\pi_k}\mathbb{P}_{\widehat{\theta_k}}(w|k)}{\sum_{k'} \widehat{\pi_{k'}}\mathbb{P}_{\widehat{\theta_{k'}}}(w|k')}$$

- Same idea than GMM!
- Bayesian variant called LDA.

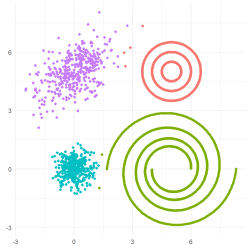
Model Based

Parametric Density Estimation Principle

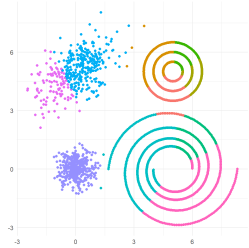
- Assign a probability of membership.
- Lots of theoretical studies...
- Model selection principle can be used to select K the number of classes (or rather to avoid using a nonsensical K...):
  - AIC / BIC / MDL penalization
  - Cross Validation is also possible!

- Complexity: $O(n \times K \times T)$

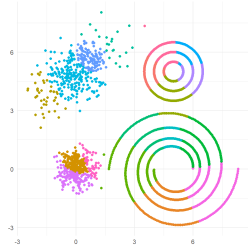
Gaussian Mixture Models



$k = 4$



$k = 10$



$k = 10$

Outline

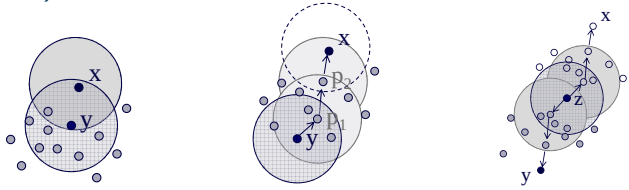
(Non Parametric) Density Based

Density Heuristic

- Cluster are connected dense zone separated by low density zone.
- Not all points belong to a cluster.

- Basic bricks:
  - Estimate the density.
  - Find points with high densities.
  - Gather those points according to the density.
- Density estimation:
  - Classical kernel density estimators...
- Gathering:
  - Link points of high density and use the resulted component.
  - Move them toward top of density *hill* by following the gradient and gather all the points arriving at the same *summit*.

(Non Parametric) Density Based

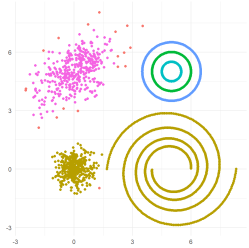


Examples

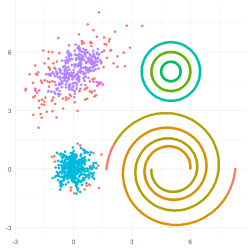
- DBSCAN: link point of high densities using a very simple kernel.
- PdfCLuster: find connected zone of high density.
- Mean-shift: move points toward top of density *hill* following an evolving kernel density estimate.

- Complexity: $O(n^2 \times T)$ in the worst case.
- Can be reduced to $O(n \log(n) T)$ if samples can be encoded in a tree structure (n-body problem type approximation).

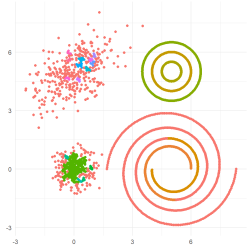
DBSCAN



$\epsilon = .45$



$\epsilon = .2$



$\epsilon = .1$

Outline

Agglomerative Clustering

Agglomerative Clustering Heuristic

- Start with very small clusters (a sample by cluster?)
- Sequential merging of the most similar clusters...
- according to some *greedy* criterion Δ.

- Generates a hierarchy of clustering instead of a single one.
- Need to select the number of cluster afterwards.
- Several choices for the merging criterion...
- Examples:
  - Minimum Linkage: merge the closest cluster in term of the usual distance
  - Ward's criterion: merge the two clusters yielding the less inner inertia loss (k-means criterion)

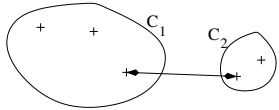
Agglomerative Clustering

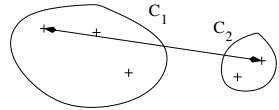
Algorithm

- Start with $(\mathcal{C}_i^{(0)}) = (\{X_i\})$ the collection of all singletons.
- At step s, we have $n - s$ clusters $(\mathcal{C}_i^{(s)})$:
  - Find the two most similar clusters according to a criterion Δ:
    $$(i, i') = \underset{(j,j')}{\operatorname{argmin}}\, \Delta(\mathcal{C}_j^{(s)}, \mathcal{C}_{j'}^{(s)})$$
  - Merge $\mathcal{C}_i^{(s)}$ and $\mathcal{C}_{i'}^{(s)}$ into $\mathcal{C}_i^{(s+1)}$
  - Keep the $n - s - 2$ other clusters $\mathcal{C}_{i''}^{(s+1)} = \mathcal{C}_{i''}^{(s)}$
- Repeat until there is only one cluster.

- Complexity: $O(n^3)$ in general.
- Can be reduced to $O(n^2)$
  - if only a bounded number of merging is possible for a given cluster,
  - for the most classical distances by maintaining a nearest neighbors list.

Agglomerative Clustering





Merging criterion based on the distance between points

- Minimum linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \min_{\underline{X}_i \in \mathcal{C}_i} \min_{\underline{X}_\in \mathcal{C}_j} d(\underline{X}_i, \underline{X}_j)$$

- Maximum linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \max_{\underline{X}_i \in \mathcal{C}_i} \max_{\underline{X}_\in \mathcal{C}_j} d(\underline{X}_i, \underline{X}_j)$$

- Average linkage:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{1}{|\mathcal{C}_i||\mathcal{C}_j|} \sum_{\underline{X}_i \in \mathcal{C}_i} \sum_{\underline{X}_\in \mathcal{C}_j} d(\underline{X}_i, \underline{X}_j)$$

- Clustering based on the proximity...

Source: E. Matzner-Löber

Agglomerative Clustering

Merging criterion based on the inertia (distance to the mean)

- Ward's criterion:

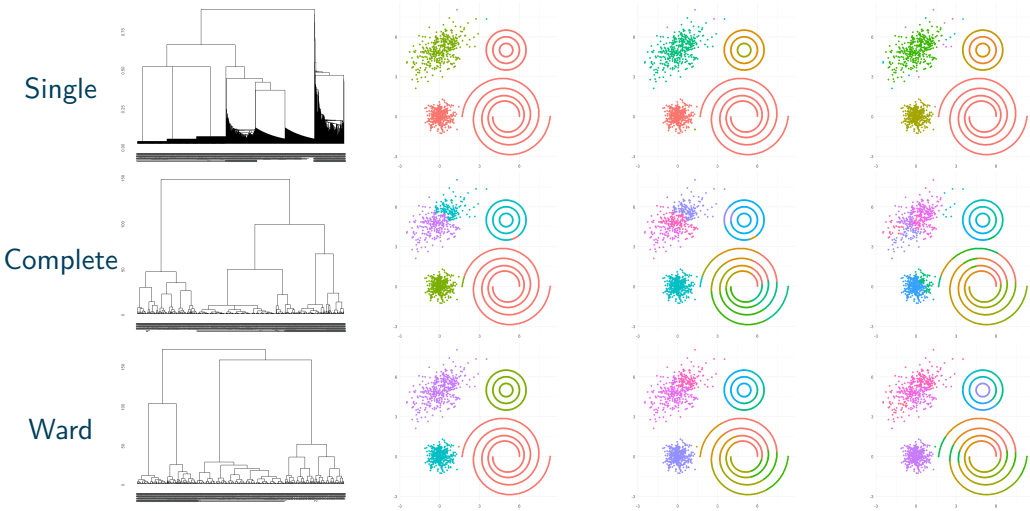
$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \sum_{\underline{X}_i \in \mathcal{C}_i} \left(d^2(\underline{X}_i, \mu_{\mathcal{C}_i \cup \mathcal{C}_j}) - d^2(\underline{X}_i, \mu_{\mathcal{C}_i}) \right) + \sum_{\underline{X}_j \in \mathcal{C}_j} \left(d^2(\underline{X}_j, \mu_{\mathcal{C}_i \cup \mathcal{C}_j}) - d^2(\underline{X}_j, \mu_{\mathcal{C}_j}) \right)$$

- If d is the Euclidean distance:

$$\Delta(\mathcal{C}_i, \mathcal{C}_j) = \frac{2|\mathcal{C}_i||\mathcal{C}_j|}{|\mathcal{C}_i| + |\mathcal{C}_j|} d^2(\mu_{\mathcal{C}_i}, \mu_{\mathcal{C}_j})$$

- Same criterion than in the k-means algorithm but greedy optimization.

Agglomerative Clustering



Outline

Grid based

Grid heuristic

- Split the space in pieces
- Group those of high density according to their proximity

- Similar to density based estimate (with partition based initial clustering)
- Space splitting can be fixed or adaptive to the data.
- Examples:
  - STING (Statistical Information Grid): Hierarchical tree construction plus DBSCAN type algorithm
  - AMR (Adaptive Mesh Refinement): Adaptive tree refinement plus k-means type assignment from high density leaves.
  - CLIQUE: Tensorial grid and 1D detection.
- Linked to Divisive clustering (DIANA)

Others

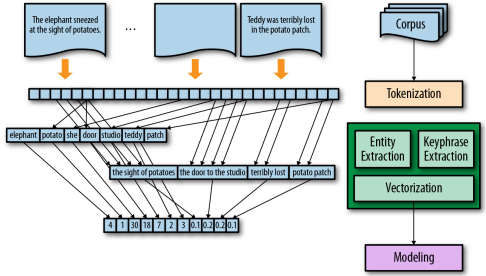
Graph based

- Spectral clustering: dimension reduction + k-means.
- Message passing: iterative local algorithm.
- Graph cut: min/max flow.

- Kohonen Map,
- ...

Outline

Text and Representation

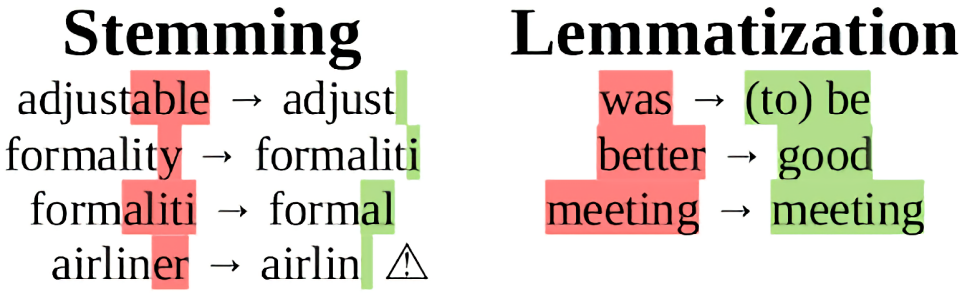


Text and Representation

- Need to transform a text into a numerical vector to reuse the previous algorithms!
- Art still in progress.

- Important steps:
  - Token extraction
  - Token vectorization
  - Learning algorithm

Source: B. Bengfort



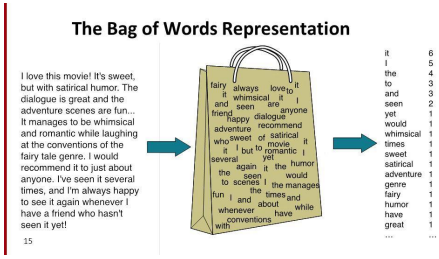
Token Extraction

- From a text to a sequence of *tokens* (words, characters, subwords...).

- Need of cleaning or pre-processing: spelling checker, stemming, lemmatization...
- Often with a further reduction of the number of possible tokens.
- Beware to not oversimplify!

Source: Quora

Bag of Words



Bag of Words

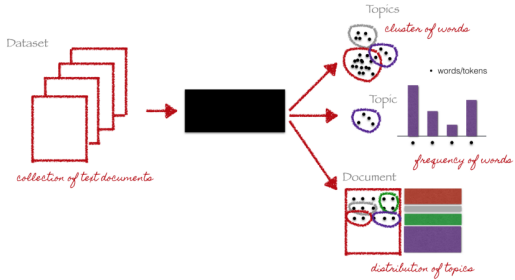
- Most simple approach to transform a text into a vector.
- Simple count of the words belonging to a predefined vocabulary.

- Counts preferably replaced by frequences (or tf-idf. . .)
- Often combined with dimension reduction:
  - restriction to an interesting vocabulary
  - use of principal component analysis (latent semantic analysis)

td-idf: text frequency - inverse document frequency

Source: Programmer Sought

Article Clustering

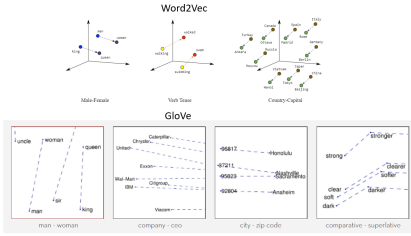


Article Clustering

- Clustering algorithms directly on the bag-of-words representation
- Most used algorithm is a variation around the k-means algorithm.

Source: C. Doig

Word Representation

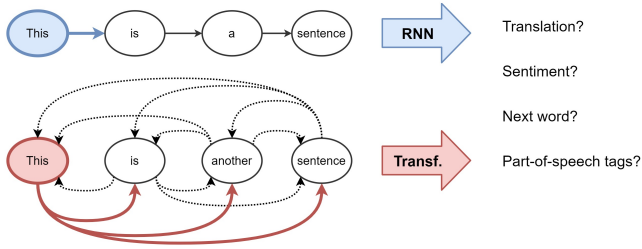


Word Representation

- More accuracy by working at the token (word) scale.
- Two approaches:
  - Associate to a word the frequency of the other words in its neighborhood and performing dimension reduction on this first representation.
  - Learn for each word a vector allowing to predict by a simple formula (scalar product) whether one word appears in the neighborhood of the other one.
- Similar results but the second approach is more flexible.

Source: R. Khandelwal

Deep Learning



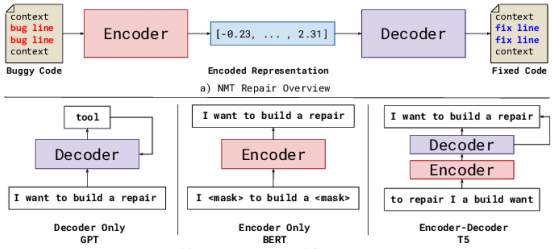
Deep Learning

- Propose a formula allowing to do computations on the word starting by associating vectors to each word.
- Learning the best possible vectors for a given task: auto-prediction (self-supervised) or prediction (supervised).

- Tremendous progress in the last years thanks to deep neural net architectures (RNN, Transformer...).

Source: Chaitanya Joshi

Large Language Models



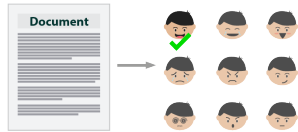
Large Language Models

- Huge neural networks relying on transformers and (pre)trained on huge corpus with self-supervised tasks.
- Three architectures:
  - Decoder: prediction of next word (online).
  - Encoder: prediction of inner word(s) (offline).
  - Encoder/Decoder: prediction of a sentence from anotherv(offline).

- Can be used as a basis for further specialized training or directly.

Source: Xia et al.

Sentiment Analysis



How to associate a sentiment to a text?

- Four possible approaches:
  - Simple approach (without learning) that averages the *sentiments* of the words used in a text using a fixed table.
  - Simple approach (supervised and linear) where this table is learned from examples.
  - Direct approach (supervised) where one predicts directly the sentiment from examples.
  - Zero-Shot approach (without learning?) where one uses directely a Large Language Model trained on a huge corpus (unrelated to the application).
- Direct approach more efficient provided one has sufficient data and one starts from a pretrained model.

Source: pngwing.com

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

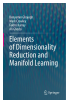
R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

References

F. Husson, S. Le, and J. Pagès.
Exploratory Multivariate Analysis by Example Using R (2nd ed.)
Chapman and Hall/CRC, 2017



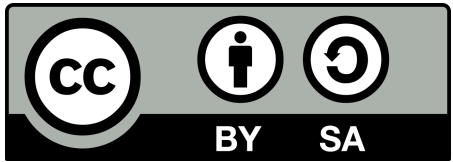
B. Ghojogh, M. Crowley, F. Karray, and A. Ghodsi.
Elements of Dimensionality Reduction and Manifold Learning.
Springer, 2023

Ch. Aggarwal and Ch. Reddy.
Data Clustering: Algorithms and Applications.
Chapman and Hall/CRC, 2013

Ch. Hennig, M. Meila, F. Murtagh, and R. Rocci.
Handbook of Cluster Analysis.
Chapman and Hall/CRC, 2015

Ch. Bouveyron, G. Celeux, B. Murphy, and A. Raftery.
Model-Based Clustering and Classification for Data Science.
Cambridge University Press, 2019

Licence and Contributors



Contributors

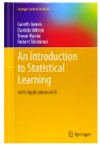
- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

Outline

Recommender Systems

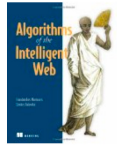
Recommended for You



An Introduction to ...
› Daniela Witten
★★★★★ (55)
~~$79.99~~ $73.58
Why recommended?

Interactive Data ...
› Scott Murray
★★★★½ (42)
~~$39.99~~ $26.85
Why recommended?

Data Smart: Using ...
› John W. Foreman
★★★★½ (66)
~~$45.00~~ $30.02
Why recommended?



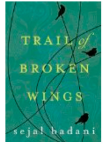
Algorithms of the ...
› H. Marmanis
★★★★☆ (16)
~~$44.99~~ $29.13
Why recommended?

Scala for Machine ...
Patrick R. Nicolas
★★★★½ (6)
~~$59.99~~ $53.99
Why recommended?

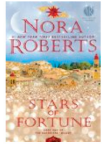
Foundations of Machine ...
› Mehryar Mohri
★★★★☆ (7)
~~$74.00~~ $66.60
Why recommended?

Hot New Releases in Kindle eBooks

New Release
The Stranger
Harlan Coben

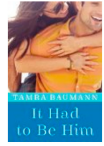


New Release
Trail of Broken Wings
Sejal Badani



Stars of Fortune: Book ...
Nora Roberts
$7.99

New Release
Boundary Crossed ...
Melissa F. Olson

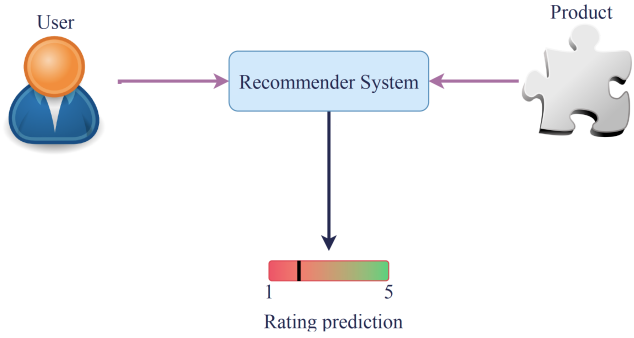


New Release
It Had to Be Him (An ...
Tamra Baumann

New Release
This Thing Called ...
Miranda Liasson

Source: Amazon

Recommender Systems



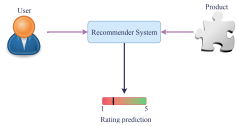
Recommender Systems

- Predict a rating for pairs of user/product,
- Use this to rank the products and suggest them to the user.

- May predict only a ranking...

Source: V. Guigue, LIP6

Data at Hands



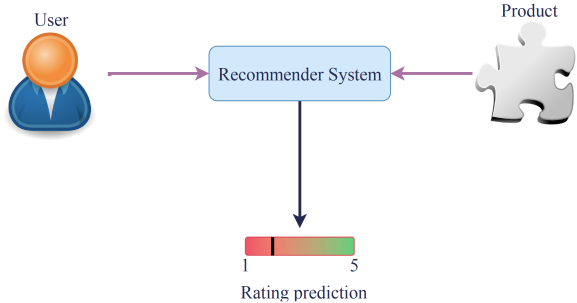
Basic observation: Triple or Pair

- Triple User/Item/Rating: (U, V, R)
- Natural interpretation as pair of User-Item/Rating: $((U, V), R)$
- Similar to the supervised setting!

Data at Hands

- Collection of pairs $((U_i, V_i), R_i)$
- User U may rate several items V and item V may be rated by several users U.
- Not in the classical i.i.d. setting because the item ratings by an user are not independent!

Goals



Goals

- Given a user U and an item V, predict the rating R.
- Rank the items V for a given user U.
- Suggest an item V to a given user U.

- We will focus on the first question!

Some Issues

User

- What is a user? An id? A detailed profile?
- What about a new user?

Item

- What is an item? An id? A detailed description? A set of features?
- What about a new item?

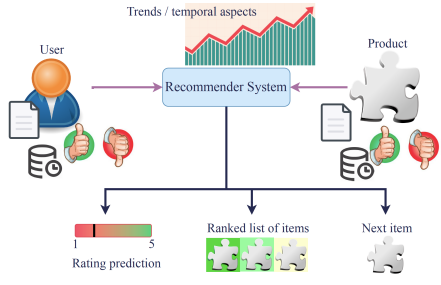
Rating

- Can we believe them?
- How to measure the error? Using the Euclidean norm?

- We will cover this...

Source: V. Guigue, LIP6

More Issues



More Issues

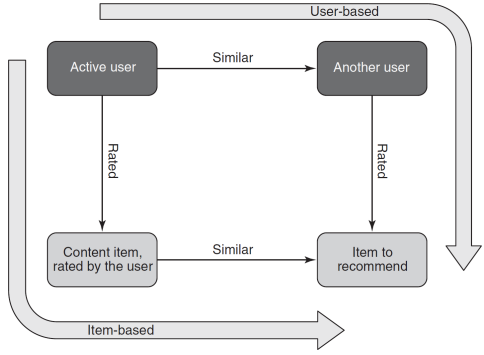
- How to take into account the temporality?
- How to take into account indirect feedbacks?
- How to propose directly a ranking?

- We won't cover that...

Source: V. Guigue, LIP6

Outline

Collaborative Filtering



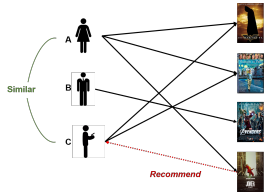
Collaborative Filtering

- Use similarity between users or items to predict ratings.

- Similar idea than in supervised learning.

Source: K. Falk

User-based Filtering



User-based Filtering

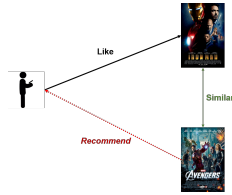
- Given a target pair of user/item (U, V).
- Choose a similarity measure $w(U, U')$ between users.
- Define a neighborhood $\mathcal{N}(U)$ of *similar* users U_i having rated V, i.e. $V_i = V$.
- Compute a predicted rating by

$$\widehat{R} = \frac{\sum_{U_i \in \mathcal{N}(U)} w(U, U_i) R_i}{\sum_{U_i \in \mathcal{N}(U)} w(U, U_i)}$$

- Choice of similarity and neighborhood will be discussed later.

Source: B. Kim

Item-based Filtering



Item-based Filtering

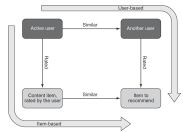
- Given a target pair of user/item (U, V).
- Choose a similarity measure $w'(V, V')$ between items.
- Define a neighborhood $\mathcal{N}(V)$ of *similar* items V_i rated by U, i.e. $U_i = U$.
- Compute a predicted rating by

$$\widehat{R} = \frac{\sum_{V_i \in \mathcal{N}'(V)} w'(V, V_i) R_i}{\sum_{V_i \in \mathcal{N}'(V)} w'(V, V_i)}$$

- Choice of similarity and neighborhood will be discussed later.

Source: B. Kim

Similarities and Neighborhood?



Similarities Based on Known Features

- Same setting than kernel density technique in supervised/unsupervised learning.

Similarities Based on Ratings

- Similarity based on (common) rated items/users.

Neighborhood

- Same setting than kernel density technique in supervised/unsupervised learning.
- Most classical approaches:
  - local – k closest neighbors or neighbors whose similarity is larger than a threshold...
  - non-local – based on a prior clustering of the users (items).

Source: K. Falk

Reminder on Similarity Measures

L^p Distance

- Formula:

$$d_p(X, X') = \left(\sum_{j=1}^{d} (X^{(j)} - X'^{(j)})^p \right)^{1/p}$$

- Renormalized version:

$$d_p(X, X') = \left(\frac{1}{d} \sum_{j=1}^{d} (X^{(j)} - X'^{(j)})^p \right)^{1/p}$$

Inverse Distance and Exponential Minus Distance

- Inverse Distance: $1/d(X, X')$
- Exponential Minus Distance: $\exp(-d(X, X'))$
- Distance may be raised to a certain power.

Reminder on Similarity Measures

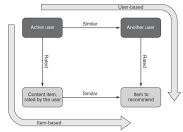
Cosine Similarity

- Formula:

$$\cos(X, X') = \frac{\sum_{j=1}^{d} X^{(j)} X'^{(j)}}{\left(\sum_{j=1}^{d} \left(X^{(j)}\right) 2\right)^{1/2} \left(\sum_{j=1}^{d} \left(X'^{(j)}\right) 2\right)^{1/2}}$$

- All those formulas require a coding of categorical variables.
- Other similarities exist!

Similarities Based on Features



Classical Features

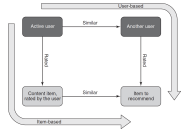
- Usual (difficult) supervised/unsupervised setting!
- (Inverse/Exponential Minus) Distance,...

Content Based Approach

- User/Item described by a text.
- NLP setting.
- Often based on a bag-of-word / keywords approach.
- (Inverse/Exponential Minus) Distance, Cosine,...

Source: K. Falk

Similarities Based on Ratings



- **Not necessarily the same number of ratings for different users or items!**

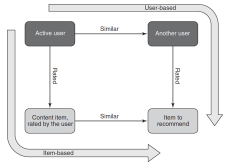
Similarity Based on Ratings

- Similarity based on the vector of rating of common rated items/rating users.
- Renormalization needed.
- (Inverse/Exponential Minus) Renormalized Distance, Cosine,...

- All the similarities can be combined...

Source: K. Falk

Local Neighborhood

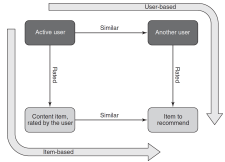


Top k / Threshold on Similarity

- Precompute the similarity for each pair of users (items) sharing an item (user)
- For any user U and item V, define the user (item) neighborhood as the k most similar users (items) sharing item V (user U) or the ones with similarity above the threshold.
- Localized neighborhood as in nearest neighbors in supervised learning.

Source: K. Falk

Non-local Neighborhood



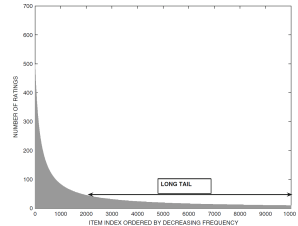
Prior Clustering

- Precompute a clustering of the users (items).
- Use the group to which user U (item V) belongs as initial neighborhood.
- Restrict it to the users (items) sharing the item V (user U)
- Non-local neighborhood as in partition based method in supervised learning.

- Strong connection with classical marketing approach!

Source: K. Falk

Ratings Issues



Ratings Issues

- User rating bias: different users may have different rating scale.
- Long tail phenomena: different users (items) may have very different number of ratings (and most users (items) have few)

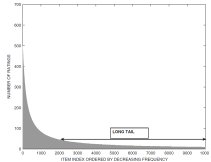
Sources: Unknown/C. Aggarwal

User Bias

User Bias

- Different users may have different rating scale.
- Possible solution:
  - Find a formula to obtain debiased ratings $D_U(R(U,V))$
  - Predict debiased rating $D_U(\widehat{R(U,V)})$ using only debiased ratings
  - Compute the biased rating using the inverse formula $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right)$
- Classical formulas:
  - Mean corrected: $D_U(R(U,V)) = R(U,V) - \overline{R(U)}$ with $\overline{R(U)}$ the mean rating for user U. so that $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right) = D(\widehat{R(U,V)}) + \overline{R(U)}$
  - Standardize: $D_U(R(U,V)) = (R(U,V) - \overline{R(U)})/\sigma(R(U))$ with $\sigma(R(U))$ the standard deviation of the ratings of user U so that $D_U^{-1}\left(D_U(\widehat{R(U,V)})\right) = \sigma(R(U))D(\widehat{R(U,V)}) + \overline{R(U)}$

Long-tail Phenomena

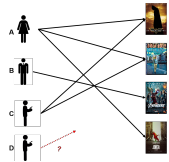


Long-tail Phenomena

- Different users/items may have very different number of ratings (and most users/items have few)
- Similarity may be biased by few items/users having a lot of ratings
- Possible solution:
  - Use a weighted similarity with a weight $-\log(N(U)/(\sum_{U'} N(U'))$ $(-\log(N(V)/(\sum_{V'} N(V'))))$ where $N(U)$ $(N(V))$ is the number of ratings of user U (item V)
- Information theory approach similar to tf-idf in NLP.

Source: C. Aggarwal

Cold Start Issue



Cold Start Issue

- Many users (items) have very few ratings.
- Some users (items) are new...

- **Not an issue for feature based or content based approaches!**

Possible Solutions

- Population approach: average based recommendation.
- Demographic approach: simple feature based recommendation.
- Scarce information approach: seeded recommendation.

Source: B. Kim

Top Items

Population Approach

- For a new user, one can use the population average to estimate $R(U, V)$
- Amount to use a constant similarity and a neighborhood equal to the whole population.
- No equivalent approach for a new item!

Demographic Approach

- If one has a *demographic* group information on the user, one may compute the average on the group.
- Amount to use a constant similarity and a neighborhood equal to the *demographic* group.
- Similar idea for a new item!

Source: B. Kim

Seeded Recommendations and Blending

Seeded Recommendations

- Compute the average on a group depending on the user behavior
- Most classical choice: compute an average on the users having given a good rating to the current viewed item
- Amount to use a constant similarity and a neighborhood equal to the group of users having given a good rating to the current viewed item.

Blending

- For user (item) with few ratings, it is often better to blend a collaborative solution with a cold start one.

Source: B. Kim

Pros and Cons

Pros

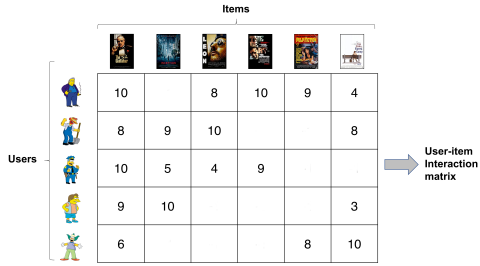
- Intuitive idea
- Easy to explain
- Can handle features and text
- Can be degraded to handle cold start

Cons

- Require an (expensive) neighborhood search!
- Require a lot of ratings to use them in similarities

Outline

Recommendation as Matrix Completion



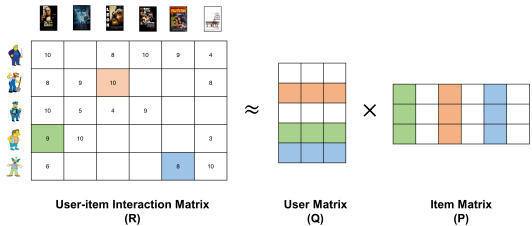
User-Item Interaction Matrix

- Matrix of ratings!
- Often most of the ratings are unknown
- Predicting the missing recommendation can be seen as completing the whole user-item interaction matrix.

- Approach based only on the ratings...

Source: B. Kim

Matrix Factorization Principle

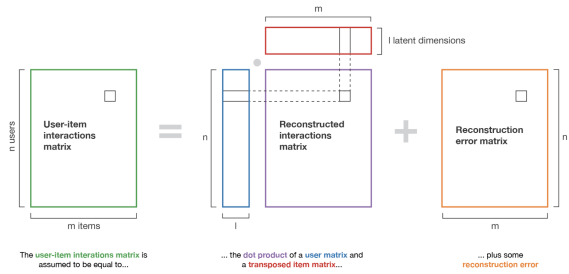


Matrix Factorization Principle

- To fill the voids, we need to add some regularity assumption.
- Simplest assumption: the $n \times p$ matrix R is (approximately) low rank, i.e $R \simeq UV^\top$ with U a $n \times k$ matrix and V a $p \times k$ matrix.

Source: B. Kim

Matrix Factorization Principle



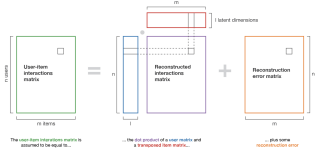
Strong Link with SVD

- Any $n \times p$ matrix $R.$ can be written $UDV^\top$ where U and V are orthogonal matrices and D is diagonal
- The best low rank approximation is obtain by restricting those matrix to the singular values with the largest eigenvalues in D.

- **Here R is not fully known so that we can't use the raw SVD!**

Source: S. Canu

Practical Factorization with SVD



SVD

- Formulation:

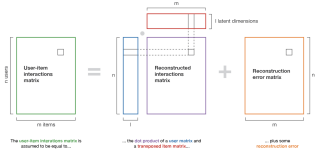
$$\underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \|R - UV^\top\|_2^2$$

$$\Leftrightarrow \underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} (R_{i,j} - U_{i,\cdot} {V_{j,\cdot}}^\top)^2$$

- Explicit solution through the SVD of the unknown R.
- May be used to obtain a baseline factorization by applying SVD to a completed R with simple replacement of the missing ratings by the mean(s).

Source: S. Canu

Practical Factorization with Weighted SVD



Weighted SVD

- Idea: Use a weight to mask the missing values in the fit
- Formulation:

$$\underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \|W \odot (R - UV^\top)\|_2^2$$

$$\Leftrightarrow \underset{U \in \mathcal{M}_{n,k}, V \in \mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2 (R_{i,j} - U_{i,\cdot} V_{j,\cdot}{}^\top)^2$$

- No explicit solution!
- Non convex optimization problem!

Source: S. Canu

Practical Factorization with Iterative Masked SVD

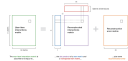
Iterative Masked SVD

- When W is a mask, i.e. $W_{i,j} \in \{0, 1\}$, there exists a simple descent algorithm!
- Algorithm:
  - Start by an initial factorization $U_0 V_0^\top$.
  - Iterate T time:
    - Compute the completed matrix $R_t = W \odot R + (1 - W) \odot (U_t V_t^\top)$
    - Use the SVD to obtain a factorization of R_t by $U_{t+1} V_{t+1}^\top$
  - Use the last factorization $U_T V_T^\top$.
- Instance of a MM algorithm without any global optimality result.
- Previous use of the SVD on the completed ratings corresponds to one step of this algorithm.

- **Computing the SVD can be very expensive!**

Source: S. Canu

Practical Factorization with Alternate Least Square



Alternate Least Square

- Weighted SVD formulation:

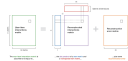
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2 \Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}.V_{j,\cdot}{}^\top)^2$$

- Optimization on U (V) corresponds to n (p) classical least-squares optimizations.
- Lead to an alternate least-squares descent algorithm without any global optimality result:
  - Start by an initial factorization $U_0 V_0{}^\top$
  - Iterate T times
    - Solve $U_{k+1} = \operatorname{argmin}_{U\in\mathcal{M}_{n,k}} \|W\odot(R-UV_k{}^\top)\|_2^2$
    - Solve $V_{k+1} = \operatorname{argmin}_{V\in\mathcal{M}_{p,k}} \|W\odot(R-U_{k+1}V^\top)\|_2^2$
  - Use $U_T V_T{}^\top$ as final factorization.

- **Computing those solutions may remain expensive!**

Source: S. Canu

Practical Factorization with SGD



Stochastic Gradient Descent

- Weighted SVD formulation:

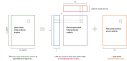
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2 \Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}V_{j,\cdot}^\top)^2$$

- Look at this problem as an optimization on $U_{i,\cdot}$ and $V_{j,\cdot}$ and use a stochastic gradient scheme without any global optimality result:
  - Start by some initial $U_{i,\cdot}$ and $V_{j,\cdot}$
  - Iterate
    - Pick uniformly a pair (i,j)
    - Update $U_{i,\cdot}$ by $U_{i,\cdot}+W_{i,j}^2\gamma(R_{i,j}-U_{i,\cdot}V_{j,\cdot}^\top)V_{j,\cdot}$
    - Update $V_{j,\cdot}$ by $V_{j,\cdot}+W_{i,j}^2\gamma(R_{i,j}-U_{i,\cdot}V_{j,\cdot}^\top)U_{i,\cdot}$
  - Use $UV^\top$ as final factorization.

- **As in any SGD scheme, the choice of the stepsize γ is very important.**

Source: S. Canu

Extension of Practical Factorization



Unbiased Rating

- Better results if one replace R with an unbiased version:
  - by subtracting the global mean (and adding it afterward)
  - by subtracting the user means (and adding them afterward)

Regularization

- Regularized Weighted SVD formulation:

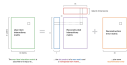
$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \|W\odot(R-UV^\top)\|_2^2+\lambda\|U\|_2^2+\lambda\|V\|_2^2$$

$$\Leftrightarrow \underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k}}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2(R_{i,j}-U_{i,\cdot}{V_{j,\cdot}}^\top)^2+\lambda\left(\sum_{i=1}^{n}\|U_{i,\cdot}\|_2^2+\sum_{j=1}^{p}\|V_{j,\cdot}\|_2^2\right)$$

- Alternate Least-Squares and SGD can be extended to this setting.

Source: S. Canu

Practical Factorization and Funk's Algorithm



Funk's Algorithm

- Funk's formulation:

$$\underset{U\in\mathcal{M}_{n,k},V\in\mathcal{M}_{p,k},\mu\in\mathbb{R},u\in\mathbb{R}^n,v\in\mathbb{R}^p}{\operatorname{argmin}} \sum_{i,j} W_{i,j}^2 (R_{i,j} - (\mu + u_i + v_j + U_{i,.} V_{j,.}^\top))^2 + \lambda \left(\mu^2 + \sum_{i=1}^{n} (u_i^2 + \| U_{i,\cdot} \|_2^2) + \sum_{j=1}^{p} (v_j^2 + \| V_{j,\cdot} \|_2^2) \right)$$

- Explicit formula including the user and item bias!
- SGD can be used in this setting!

- **Lead to state of the art results!**

Source: S. Canu

Pros and Cons

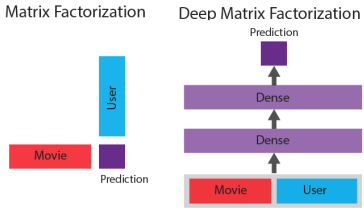
Pros

- Quite efficient even if the rating matrix is sparse.
- Lead to an explicit formula for any pair of user/item.
- Efficient numerical algorithm.

Cons

- No straightforward explanation of the prediction.
- Do not use features or text.
- No way to handle cold start.

Recommendation as Prediction



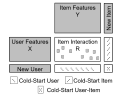
Factorization as a Prediction Algorithm

- Optimization of a formula
$$R(U_i, V_j) = \mu + u_i + v_j + U_{i,\cdot} V_{j,\cdot}^{\top}$$
with a least-squares criterion.
- Other formulas are probably possible...
- Key: representation learning ? Can we use Deep Learning?

- **Not easy to do better than matrix factorization with a classical DNN!**
- **Explicit scalar product seems required!**

Source: J. Schreiber

Model Based Recommendation

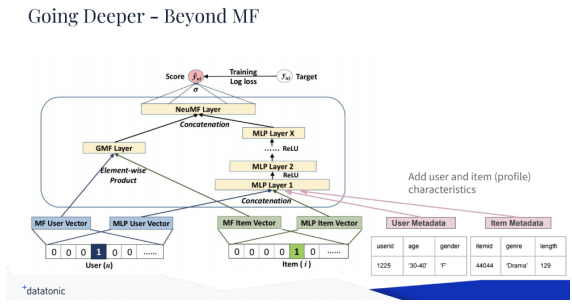


Model Based Recommandation

- Optimization of a formula:
$$R(U_i, V_j) = f(U_i, V_j)$$
where U_i and V_i can be a combination of an id (one hot encoding) and features.
- Simple additive models:
$$R(U_i, V_i) = f_U(U_i) + f_V(V_j)$$
- Models with explicit interactions:
$$R(U_i, V_i) = f_U(U_i) + f_V(V_j) + F_{UV}(U_i, V_j)$$
- Possible extension of factorization based on
$$F_{UV}(U_i, V_i) = M_U U_i (M_v V_j)^\top$$
- Link with transformers...

Source: DeepAI

Deep Recommendation



Deep Recommendation

- Combine an explicit dot product structure with a classical DNN.
- Allow to learn a representation and to add features / text content directly.
- Large flexibility in the architecture.

Source: Altoros

Pros and Cons

Pros

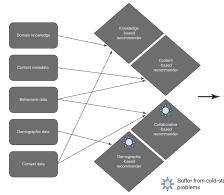
- Combine the strength of the factorization based and the feature based methods
- Best performances...

Cons

- Not so easy to construct a good formula/architecture...
- Not so easy to train...
- Not easy to beat raw matrix factorization (when using only user/item interactions)!

Outline

Hybrid Recommender



Hybrid Recommender

- Combine the scores of several recommendation algorithms.
- Can be casted as an ensemble method where the number of interactions is used.

Pros

- Lots of flexibility

Cons

- Lots of flexibility!

Source: K. Falk

Performance Measure

CASE 1: Evenly distributed errors

ID	Error	|Error|	Error^2
1	2	2	4
2	2	2	4
3	2	2	4
4	2	2	4
5	2	2	4
6	2	2	4
7	2	2	4
8	2	2	4
9	2	2	4
10	2	2	4

MAE	RMSE
2.000	2.000

CASE 2: Small variance in errors

ID	Error	|Error|	Error^2
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	3	3	9
7	3	3	9
8	3	3	9
9	3	3	9
10	3	3	9

MAE	RMSE
2.000	2.236

CASE 3: Large error outlier

ID	Error	|Error|	Error^2
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	20	20	400

MAE	RMSE
2.000	6.325

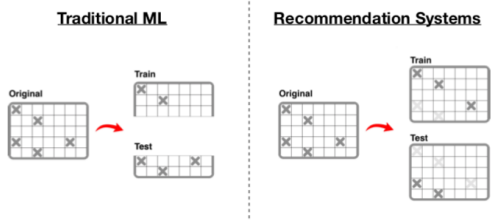
- **Need of a metric to measure the performance!**

Metric on the ratings

- RMSE:
  - Most classical choice
  - Implicitly used in collaborative filtering and explicitly in matrix factorization.
  - Easy to use.
- MAE: more robust to outliers...

Source: JJ

Validation



- **Need of validation technique!**

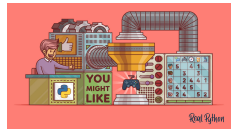
Validation Scheme

- Much more complicated that the usual supervised setting.
- Lack of independence of the observations.
- Most classical choice: random partition of the ratings!

- No strong theoretical support!

Source: J. Cates

Metric vs Goals



- Are those metrics really the right thing to optimize?

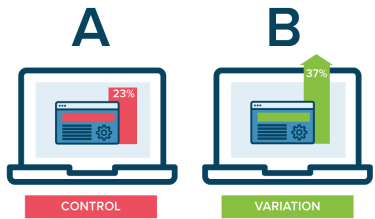
Better Goals

- Diversity : do not always suggest the same items.
- Coverage: suggest most of the items to at least some users.
- Serendipity: suggest **surprising** items.
- Business Goal: Sell more! Earn more money!

- Explain why there is a lot of post-processing to go from the ratings to the suggested item list!
- For instance: use of lift instead of ratings, use of localization, use of randomization...

Source: Real Python

A/B Testing



A/B Testing

- No direct way to estimate the performance according to non trivial metric.
- Solution: perform experiment to test whether a method is good or not!
- A/B Testing: classical hypothesis testing on the means (or the proportions).
- Bandit approach: real-time optimization of the allocation (not much used in practice).

Source: Optimizely

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

References

K. Falk.
Practical Recommender Systems.
Manning, 2019

F. Ricci, L. Rokach, and B. Shapira.
Recommender Systems Handbook (3rd ed.)
Springer, 2022

Ch. Aggarwal.
Recommender Systems, The Textbook.
Springer, 2016

Outline

Outline

Text and Bag of Words

- How to transform a **text** into a vector of **numerical features**?

Bag of Words strategy

- Make a **list** of words.
- Compute a **weight** for each word.

List building

- Make the list of all used words with their number of occurrence.
- Gather the words in the list having the same stem (stemming).
- Compute the histogram $h_w(d)$.

Text and Bag of Words

Weight computation

- Apply a renormalization:
  - tf transform (word profile):
    $$\mathrm{tf}_w(d) = \frac{h_w(d)}{\sum_w h_w(d)}$$
    so that $\mathrm{tf}_w(d)$ is the frequency within the document d.
  - tf-idf transform (word profile weighted by rarity):
    $$\mathrm{tf-idf}_w(d) = \mathrm{idf}_w \times \mathrm{tf}_w(d)$$
    with idf a corpus dependent weight
    $$\mathrm{idf}_w = \log \frac{n}{\sum_{i=1}^{n} \mathbf{1}_{h_w(d_i)\neq 0}}$$

- Use the vector $\mathrm{tf}(d)$ (or $\mathrm{tf-idf}(d)$) to describe a document.
- Most classical text preprocessing!
- Latent Semantic Analysis: PCA of this representation.
- Hashing can be used to reduce the number of words.

Stemming and Lemmatization

Stemming

adjustable → adjust
formality → formaliti
formaliti → formal
airliner → airlin ⚠

Lemmatization

was → (to) be
better → good
meeting → meeting

Text Preprocessing

- Very important step in text processing.
- Art of obtaining good tokens.
- Spelling correction, Stemming, Lemmatization...

Source: Quora

Okapi BM25 for Text Retrieval

Okapi BM25

- Representation (smoothed tf-idf):

$$\mathrm{bm25}_w(d) = \mathrm{idf}_w \times \frac{(k_1+1)\mathrm{tf}_w(d)}{k_1 + \mathrm{tf}_w(d)}$$

- Match quality for a set of words Q measured by a simple scalar product:

$$\mathrm{BM25}(d, Q) = \sum_{w \in Q} \mathrm{bm25}_w(d)$$

- Extensively used in text retrieval.
- Can be traced back to 1976!

Probabilistic latent semantic analysis (PLSA)

- Model:

$$\mathbb{P}(\mathrm{tf}) = \sum_{k=1}^{K} \mathbb{P}(k)\,\mathbb{P}(\mathrm{tf}|k)$$

  with k the (hidden) topic, $\mathbb{P}(k)$ a topic probability and $\mathbb{P}(\mathrm{tf}|k)$ a multinomial law for a given topic.

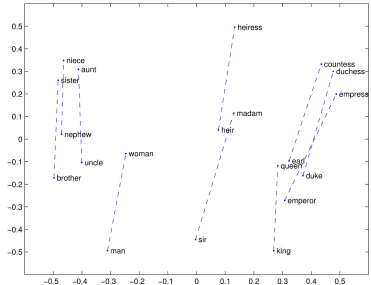
- Clustering according to a mixture model

$$\mathbb{P}(k|\mathrm{tf}) = \frac{\widehat{\mathbb{P}(k)}\widehat{\mathbb{P}(\mathrm{tf}|k)}}{\sum_{k'} \widehat{\mathbb{P}(k')}\widehat{\mathbb{P}(\mathrm{tf}|k')}}$$

- Same idea than GMM!
- Bayesian variant called LDA.

Outline

Word Vectors



Word Embedding

- Map from the set of words to $\mathbb{R}^d$.
- Each word is associated to a vector.
- Hope that the relationship between two vectors is related to the relationship between the corresponding words!

Source: nlp.stanford.edu

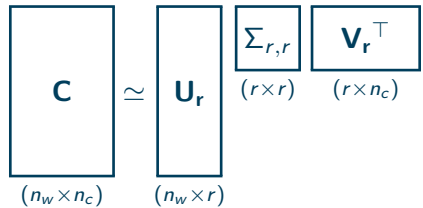
Look ! A single **word** and its context

Word And Context

- **Idea:** characterize a word w through its relation with words c appearing in its context...
- **Probabilistic description**:
  - Joint distribution: $f(w,c) = \mathbb{P}(w,c)$
  - Conditional distribution(s): $f(w,c) = \mathbb{P}(w|c)$ or $f(w,c) = \mathbb{P}(c|w)$.
  - Pointwise mutual information: $f(w,c) = \mathbb{P}(w,c)/(\mathbb{P}(w)\mathbb{P}(c))$
- Word w characterized by the vector $C_w = (f(w,c))_c$ or $C_w = (\log f(w,c))_c$.

- In practice, C is replaced by an estimate on large corpus.
- Very high dimensional model!

A (Naïve) SVD Approach



Truncated SVD Approach

- Approximate the embedding matrix C using the truncated SVD decomposition (best low rank approximation).
- Use as a code

$$C'_w = U_{r,w}\Sigma^{\alpha}_{r,r}$$

with $\alpha \in [0, 1]$.

- Variation possible on C.
- State of the art results but computationally intensive...

A Least-Squares Approach

- All the previous models correspond to
$$-log\mathbb{P}(w,c) \sim C_w'^t C_c'' + \alpha_w + \beta_c$$

GloVe (Global Vectors)

- Enforce such a fit through a (weighted) least-squares formulation:
$$\sum_{w,c} h(\mathbb{P}(w,c)) \left\| -log\mathbb{P}(w,c) - (C_w'^t C_c'' + \alpha_w + \beta_c) \right\|^2$$
with h a increasing weight.
- Minimization by alternating least square or stochastic gradient descent...

- Much more efficient than SVD.
- Similar idea in recommendation system.

A Learning Approach

Supervised Learning Formulation

- True pairs (w, c) are positive examples.
- Artificially generate negative examples (w', c') (for instance by drawing c' and w' independently in the same corpus.)
- Model the probability of being a true pair (w, c) as a (simple) function of the codes C'_w and C''_c.

- Word2vec: logistic modeling

$$\mathbb{P}(1|w, c) = \frac{e^{C'^t_w C''_c}}{1 + e^{C'^t_w C''_c}}$$

- State of the art and efficient computation.
- Similar to a factorization of $-\log(\mathbb{P}(w, c)/(\mathbb{P}(w)\mathbb{P}(c)))$ but without requiring the estimation of the probabilities!

Outline

Text as Sequences

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. This creates an internal state of the network which allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented connected handwriting recognition or speech recognition.

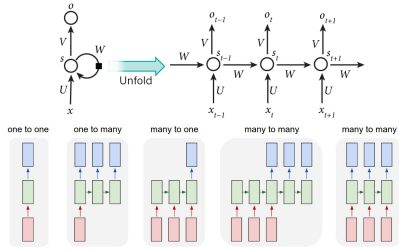
Sequences

- Word = sequence of letters.
- Text = sequence of letters/words.

- Capitalize on this structure.

Source: Wikipedia

Recurrent Neural Networks



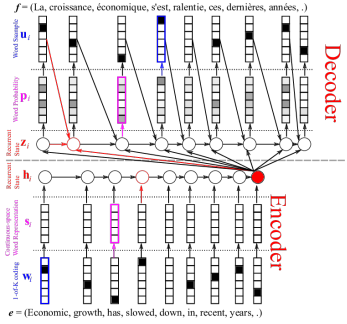
Recurrent Neural Network Unit

- Input seen as a sequence.
- **Simple** computational units with shared weights.
- Information transfer through a context!

- Several architectures!

Sources: Y. LeCun / Unknown

Automatic Translation

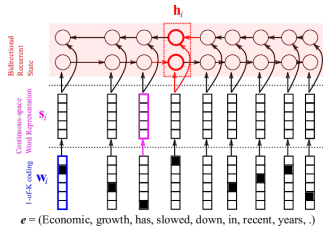


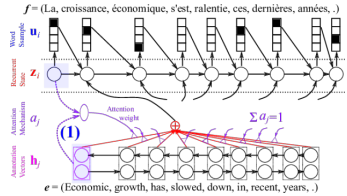
Encoder/Decoder structure

- Word vectors, RNN, stacked structure.

Source: Nvidia

Automatic Translation



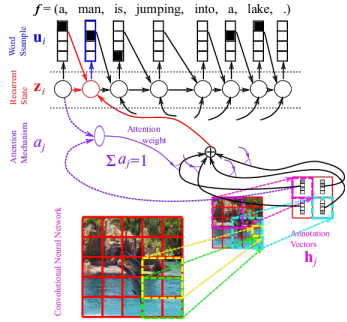


Encoder/Decoder structure

- Much more complex structure: asymmetric, attention order...

Source: Nvidia

Automatic Captioning

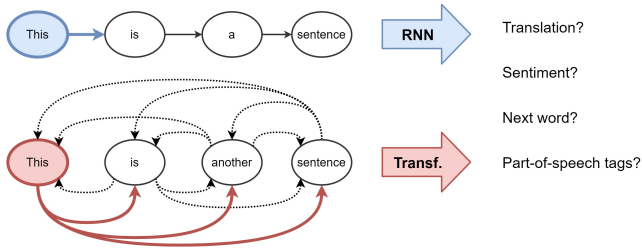


Encoder/Decoder structure

- Much more complex structure: asymmetric, attention order...

Source: Nvidia

Text as Graph

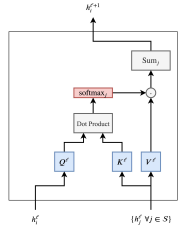


Text as Graph

- More than just sequential dependency.
- Each word is related to (all the) other words.
- Graph structure with words and directed relations between words.

Source: Chaitanya Joshi

Attention



Attention between words

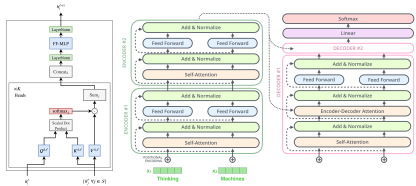
- Words encoded by h_i at layer l.
- Compute individual value for each word: $v_i = V^l h_i$
- Compute combined value for each word: $h_i' = \sum_j w_{i,j} v_j$
- **(Self) Attention:** weight $w_{i,j}$ defined by

$$w_{i,j} = \text{SoftMax}\left(\left\langle Q^l h_i, K^l h_j \right\rangle\right)$$

- $Q^l h_i$ is called a query and $K^l h_j$ a key.

Source: Chaitanya Joshi

Transformer



Transformer

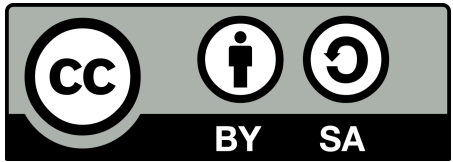
- Block combining several attention heads and a classical MLP.

Encoder/Decoder Architecture

- Combine several transformers and more MLP in a task-adapted architecture.
- End-to-end training is not easy (initialization, optimization...).
- Initial embedding at token level rather than word level to cope with new words!

Sources: Chaitanya Joshi / Jay Alamar

Licence and Contributors



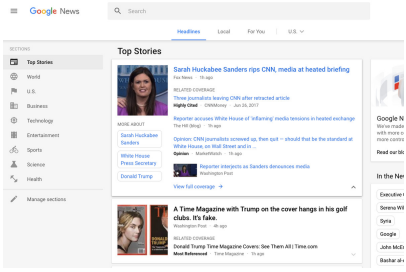
Contributors

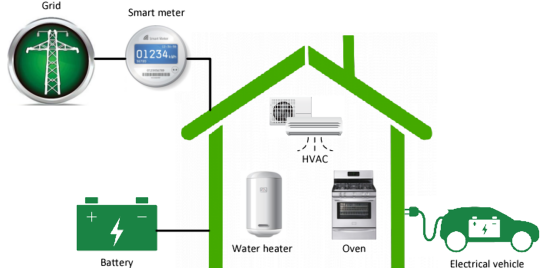
- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

Outline

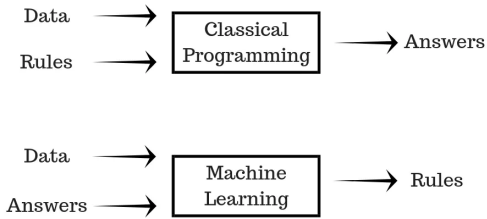
Machine Learning





Sources: MyCarDoesWhat.org/theverge.com/Zhiqiang Wan et al.

Machine Learning



A definition by Tom Mitchell

A computer program is said to learn from **experience E** with respect to some **class of tasks T** and **performance measure P**, if its performance at tasks in T, as measured by P, improves with experience E.

Source: C. Barabasi

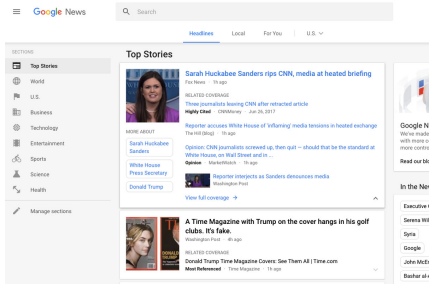
Object Detection

A detection algorithm:

- **Task**: say if an object is present or not in the image
- **Performance**: number of errors
- **Experience**: set of previously seen labeled images

Source: MyCarDoesWhat.org

Article Clustering

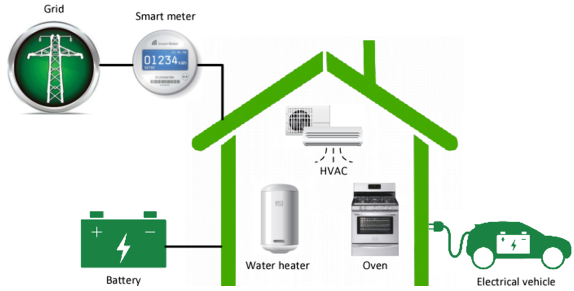


An article clustering algorithm:

- **Task**: group articles corresponding to the same news
- **Performance**: quality of the clusters
- **Experience**: set of articles

Source: theverge.com

Smart Grid Controler

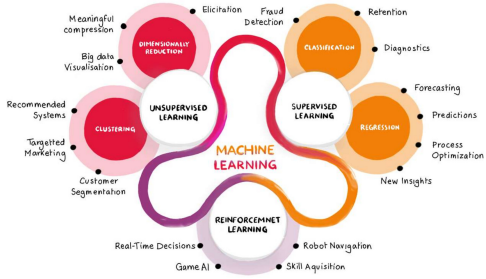


A controler in its sensors in a home smart grid:

- **Task**: control the devices
- **Performance**: energy costs
- **Experience**:
  - previous days
  - current environment and performed actions

Source: Zhiqiang Wan et al.

Three Kinds of Learning

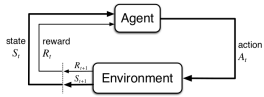


Unsupervised Learning	Supervised Learning	Reinforcement Learning
Task: Clustering/DR/Generative	**Task:** Prediction/Classification	**Task:** Actions
Performance: Quality	**Performance:** Average error	**Performance:** Total reward
Experience: Raw dataset (No Ground Truth)	**Experience:** Good Predictions (Ground Truth)	**Experience:** Reward from env. (Interact. with env.)

- **Timing:** Offline/Batch (learning from past data) vs Online (continuous learning)

Source: BCG

Reinforcement Learning



Reinforcement Learning Setting

- **Env.:** provides a reward and a new state for any action.
- **Agent policy π:** choice of an action A_t from the state S_t.
- **Total reward:** (discounted) sum of the rewards.

Questions

- **Policy evaluation:** how to evaluate the expected reward of a policy knowing the environment?
- **Planning:** how to find the best policy knowing the environment?
- **Reinforcement Learning:** how to find the best policy without knowing the environment?

Outline

Decision or Decisions

Source: W. Powell

Sequential Decision Setting

Sequential Decision Setting

- In many (most?) settings, not a single decision but a sequence of decisions.
- Need to take into account the (not necessarily immediate) consequences of the sequence of decisions/actions rather than of each decisions.
- Different framework than supervised learning (no immediate feedback here) and unsupervised learning (well defined goal here).

Source: W. Powell

From Sequential Decision to Reinforcement Learning

Sequential Decision

Sequential Decision

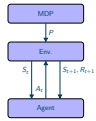
- Sequence of action A_t as a response of an environment S_t
- Feedback through a reward R_t

Actions?

- Is my current way of choosing actions good?
- How to make it better?

From Sequential Decision to Reinforcement Learning

Sequential Decision



MDP Modeling

Markov Decision Process Modeling

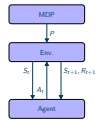
- Specific modeling of the environment.
- Goal as as a (weighted) sum of a scalar reward.

Actions?

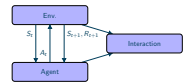
- Is my current way of choosing actions good?
- How to make it better?

From Sequential Decision to Reinforcement Learning

Sequential Decision



MDP Modeling



Reinforcement Learning

Reinforcement Learning

- Same modeling...
- But no direct knowledge of the MDP.

Actions?

- Is my current way of choosing actions good?
- How to make it better?

Sequential Decision Settings

- MDP / Reinforcement Learning:

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{t} R_t\right]$$

- Optimal Control:

$$\min_{u} \mathbb{E}\left[\sum_{t} C(x_t, u_t)\right]$$

- (Stochastic) Search:

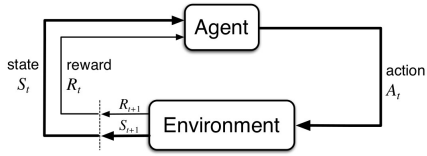
$$\max_{\theta} \mathbb{E}[F(\theta, W)]$$

- Online Regret:

$$\max \sum_{k} \mathbb{E}[F(\theta_k, W)]$$

Outline

The Agent-Environment Interface



Markovian Decision Processes

- At time step $t \in \mathbb{N}$:
  - State $S_t \in \mathcal{S}$: representation of the environment
  - Action $A_t \in \mathcal{A}(S_t)$: action chosen
  - Reward $R_{t+1} \in \mathcal{R}$: instantaneous real valued reward
  - New state S_{t+1}
- Main assumption: dynamic entirely defined by

$$\mathbb{P}(S_{t+1} = s', R_{t+1} = r | S_t = s, A_t = a) = p(s', r | s, a)$$

- Finite MDP: $\mathcal{S}$, $\mathcal{A}$ and $\mathcal{R}$ are finite.

Returns and Episodes

Return

- (Discounted) Return:

$$G_t = \sum_{t'=t+1}^{T} \gamma^{t'-(t+1)} R_{t'}$$

- Finiteness if $|R| \leq M$

$$|G_t| \leq \begin{cases} (T-(t+1))M & \text{if } T < \infty \\ M\frac{1}{1-\gamma} & \text{otherwise} \end{cases}$$

- Not well-defined if $T = \infty$ and $\gamma = 1$.
- Recursive property

$$G_t = R_{t+1} + \gamma G_{t+1}$$

Policies and Value Functions

Policy and Value Functions

- Policy: $\pi(a|s)$
- Value function:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right]$$

- Action value function:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a]$$

Two natural problems

- Policy evaluation: compute v_π given π.
- Planning: find $\pi^\star$ such that $v_{\pi^\star}(s) \geq v_\pi(s)$ for all s and π.

- Those objects may not exist in general!
- Can be traced back to the 50s!

MDP vs Discrete Control

MDP

- State s and action a
- Dynamic model:
$$\mathbb{P}(s'|s,a)$$
- Reward r defined by $\mathbb{P}(r|s',s,a)$.
- Policy Π: $a_t = \pi_t(S_t, H_t)$
- Goal:
$$\max \mathbb{E}_\Pi \left[\sum_t R_t \right]$$

Discrete Control

- State x and control u
- Dynamic model:
$$x' = f(x,u,W)$$
with W a stochastic perturbation.
- Cost: $C(x,u,W)$.
- Control strategy U: $u_t = u(x_t, H_t)$
- Goal:
$$\min_U \mathbb{E}_U \left[\sum_t C(x_t, u_t, W_t) \right]$$

- Almost the same setting but with a different vocabulary!

Outline

Policy Evaluation by Bellman Backup

Fixed Point Property

- Bellman Equation

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma v_\pi(s') \right] = \mathcal{T}_\pi(v_\pi)(s)$$

- Linear equation that can be solved.

Policy Evaluation by Dynamic Programming

- Fixed point iterative algorithm: $v_{k+1}(s) = \mathcal{T}_\pi(v_k)(s)$

- Converge if $T < \infty$ or $\gamma < 1$.

Planning by Policy Improvement

Policy Improvement Property

- If π' is such that $\forall s, q_\pi(s, \pi'(s)) \geq v_\pi(s)$ then $v_{\pi'} \geq v_\pi$.

Policy Iteration Algorithm

- Compute v_{π_k}
- Greedy update:

$$\begin{aligned}\pi_{k+1}(s) &= \operatorname*{argmax}_a q_{\pi_k}(s, a) \\ &= \operatorname*{argmax}_a \sum_{s', r} p(s', r|s, a) \left(r + \gamma v_{\pi_k}(s')\right)\end{aligned}$$

- If $\pi' = \pi$ after a greedy update $v_{\pi_{k+1}} = v_{\pi_k} = v_*$.

- Convergence in finite time in the finite setting.
- ϵ-greedy improvement among ϵ-policy: classical improvement degraded by picking uniformly the action with probability ϵ

Planning by Bellman Backup

Fixed Point Property

- Bellman Equation

$$v_*(s) = \max_a \sum_{s'} \sum_r p(s', r|s, a) \left[r + \gamma v_*(s') \right] = \mathcal{T}_*(v_*)(s)$$

- Linear programming problem that can be solved.

Policy Evaluation by Dynamic Programming

- Iterative algorithm: $v_{k+1}(s) = \mathcal{T}_*(v_k)(s)$

- Converge if $T < \infty$ or $\gamma < 1$.
- Amount to improve the policy after only one step of policy evaluation.

Planning by Bellman Backup

Q-value and enhancement

- Q-value:

$$q_\pi(s,a) = \sum_{s'} \sum_{r} p(s',r|s,a) \left[r + \gamma \sum_{a'} \pi(a'|s') q_\pi(s',a') \right]$$

- Easy policy enhancement: $\pi'(s) = \operatorname*{argmax}_a q_\pi(s,a)$

Fixed Point Property

- Bellman Equation

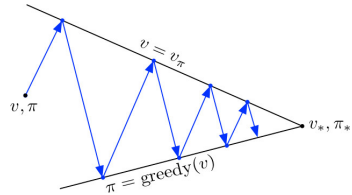
$$q_*(s,a) = \sum_{s'} \sum_{r} p(s',r|s,a) \left[r + \gamma \max_{a'} q_*(s',a') \right] = \mathcal{T}_*(q_*)(s,a)$$

- Linear programming problem that can be solved.

Policy Evaluation by Dynamic Programming

- Iterative algorithm: $q_{k+1}(s,a) = \mathcal{T}_*(q_k)(s,a)$

Generalized Policy Iteration



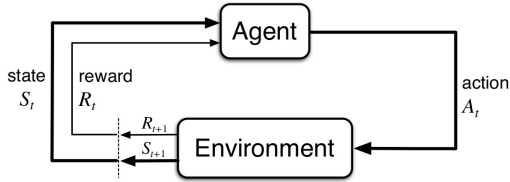
Generalized Policy Iteration

- Consists of two simultaneous interacting processes:
  - one making a value function consistent with the current policy (policy evaluation)
  - one making the policy greedy with respect to the current value function (policy improvement)

- Stabilizes only if one reaches the optimal value/policy pair.
- Asynchronous update are possible provided every state(/action) is visited infinitely often.
- Very efficient but requires the knowledge of the transition probabilities.

Outline

Reinforcement Learning



Reinforcement Learning - Sutton (98)

- An agent takes actions in a sequential way, receives rewards from the environment and tries to maximize his long-term (cumulative) reward.

Reinforcement Learning

- MDP setting with cumulative reward.
- Planning problem.
- Environment known only through interaction, i.e. some sequences $\cdots S_t A_t R_{t+1} S_{t+1} A_{t+1} \cdots$.

RL: More than planning?

Prediction

- Known π and access to interactions with MDP and estimation of v_π.

Planning

- Access to interactions with MDP and estimation of a good (optimal?) policy π.

Imitation Learning

- Observation of interactions with an unknown policy and estimation of this policy.
- Back to Supervised Learning setting.

Inverse Reinforcement Learning

- Observation of interactions following a policy π and estimation of rewards so that this (implicitly Gibbs type) policy is (almost) optimal.

- Focus on prediction/planning!

Monte Carlo

MC Methods

- Back to $v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$.
- Monte Carlo:
  - Play several episodes using policy π.
  - Average the returns obtained after any state s.
- Online algorithm: $V(S_t) \leftarrow V(S_t) + \alpha(G_t - V(S_t))$.

- Good theoretical properties provided every states are visited asymptotically *infinitely often*.

Extensions

- Off-policy setting (behavior policy $b \neq$ target policy π) with importance sampling.
- Planning with policy improvement steps (estimating q_π instead of v_π)

- No theoretical results for the last case.
- Need to wait until the end of an episode to update anything...

Bootstrap and TD Prediction

Bootstrap and TD

- Bootstrap idea: Replace G_t by $R_{t+1} + \gamma v_\pi(S_{t+1})$.
- Temporal Difference: stochastic approximation scheme
$$V(S_t) \leftarrow V(S_t) + \alpha \left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$
- Update occurs at each time step.
- Rely on
$$\begin{aligned} v_\pi(s) &= \mathcal{T}_\pi v_\pi(s) \\ &= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \end{aligned}$$
- Can be proved to converge (under some assumption on α).

- Combine the best of Dynamic Programing and MC.
- Can be written in term of Q:
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)\right)$$

SARSA and Q Learning

- How to use this principle to obtain the best policy?

SARSA: Planning by Prediction and Improvement (online)

- Update Q following the current policy π

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right)$$

- Update π by policy improvement.

- May not converge if one use a greedy policy update

Q Learning: Planning by Bellman Backup (off-line)

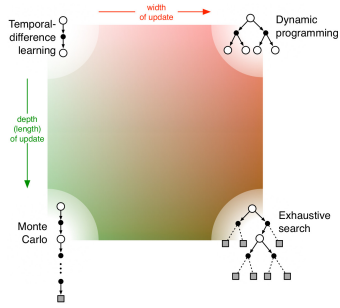
- Update Q following the behavior policy b

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right)$$

- No need to use importance sampling correction for depth 1 update.
- Final policy deduced from Q.

- Proof of convergence in both cases.

Variations



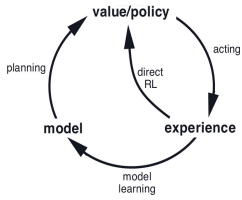
Depth

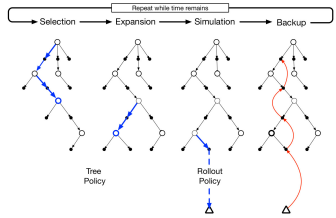
- Number of steps in the update.

Width

- Number of states/actions considered at each step.

Planning and Learning





Planning and Models

- Planning can combine model estimation (DP) and direct learning (RL).

Real Time Planning

- Planning can be made online starting from the current state.

- Curse of dimensionality: methods are hard to use when the cardinality of the states and the actions are large!

Outline

Value Function Approximation

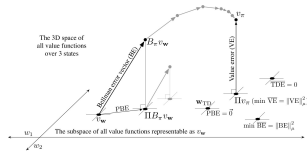
Value Function Approximation

- **Idea:** replace $v(s)$ by a parametric $\hat{v}(s, \boldsymbol{w})$.
- **Issues:**
  - Which approximation functions?
  - How to define the quality of the approximation?
  - How to estimate $\boldsymbol{w}$?

Approximation functions

- Any parametric (or kernel based) approximation could be used.
- Most classical choice:
  - Linear approximation.
  - Deep Neural Nets...

Approximation Quality



- How define when $\hat{v}(\cdot, \boldsymbol{w})$ is close to v_π (or v_*)

Prediction(/Control)

- Prediction objective:
$$\sum_s \mu(s)(v_\pi(s) - \hat{v}(s, \boldsymbol{w}))^2$$
- Bellman Residual:
$$\sum_s \mu(s)(\mathcal{T}_\pi \hat{v}(s, \boldsymbol{w}) - \hat{v}(s, \boldsymbol{w}))^2$$

  or its projection...

- **Issues:**
  - Neither v_π nor $\mathcal{T}_\pi$ are known...
  - No connection between a policy associated to $\hat{v}$ and π...

Online Gradient and Semi-Gradient

Online Prediction

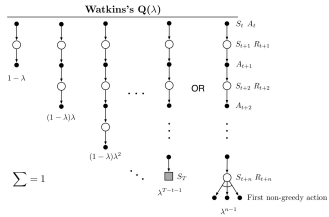
- SGD algorithm on $\boldsymbol{w}$:
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(v_\pi(S_t) - \hat{v}(S_t, \boldsymbol{w}_t) \right) \nabla \hat{v}(S_t, \boldsymbol{w}_t)$$
- MC approximation (still SGD):
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(G_t - \hat{v}(S_t, \boldsymbol{w}_t) \right) \nabla \hat{v}(S_t, \boldsymbol{w}_t)$$
- TD approximation (not SGD anymore):
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{v}(S_t, \boldsymbol{w}_t) \right) \nabla \hat{v}(S_t, \boldsymbol{w}_t)$$

- Deeper or wider scheme possible.

Online Control

- SARSA-like algorithm:
  - Prediction step as previously with the current policy
  $$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \hat{q}(S_{t+1}, A_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}) \right) \nabla \hat{q}(S_t, A_t, \boldsymbol{w}_t)$$
  - ϵ-greedy update of the current policy

Offline Control with Approximation



Offline Control

- Q-Learning like algorithm:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \left(R_{t+1} + \gamma \max_a \hat{q}(S_{t+1}, a, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t) \right) \times \nabla \hat{q}(S_t, A_t, \boldsymbol{w}_t)$$

  with an arbitrary policy b.
- Deeper formulation using importance sampling possible.

- **Issue:** Hard to make it converge in general!

Deadly Triad

Sutton-Barto's Deadly Triad

- **Function Approximation**
- **Bootstrapping**
- **Off-policy training**

Deep Q-Learning Stabilization Tricks

- Memory replay: sample from a set of episodes (sampled model)
- Frozen Q:
  - use previous weights in the max (two-scales algorithms)
  - Amount to an approximate value iteration algorithm!

- Good mathematical heuristics lead to better practical results!

Policy Based Approach

- Other approach with a **parametric policy**.

Parametric Policy Setting

- New goal:

$$J(\theta) = \sum_s \mu_{\pi_\theta}(s) v_{\pi_\theta}(s)$$
$$= \sum_s \mu_{\pi_\theta}(s) \sum_a \pi_\theta(a|s) q_{\pi_\theta}(s,a)$$

- Stochastic gradient:

$$\widehat{\nabla} J(\theta) = \sum_t \gamma^t \nabla \log \pi_\theta(A_t|S_t)(q_{\pi_\theta}(S_T, A_T) - v_{\pi_\theta}(S_t))$$
$$\sum_t \gamma^t \nabla \log \pi_\theta(A_t|S_t) a_{\pi_\theta}(S_T, A_T)$$

- On policy algorithm if we can estimate $a_{\pi_\theta}(S_T, A_T) = q_{\pi_\theta}(S_T, A_T) - v_{\pi_\theta}(S_t)$ for instance by MC.

Actor-Critic

- Simultaneous parameterization of
  - the policy π by $\boldsymbol{\theta}$,
  - the value function Q (and $V(s) = \mathbb{E}_\pi[Q(s, \cdot)]$) by $\boldsymbol{w}$
- Simultaneous update:

$$\delta_t = R_t + \gamma \hat{v}(S_{t+1}, \boldsymbol{w}_t) - \hat{q}(S_t, A_t, \boldsymbol{w}_t)$$
$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha \delta_t \nabla \hat{q}(S_t, A_t, \boldsymbol{w}_t)$$
$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \beta \left(Q_{\boldsymbol{w}}(S_t, A_t) - V_{\boldsymbol{w}}(S_t)\right) \nabla \log \pi_{\boldsymbol{\theta}}(a|S_t, \boldsymbol{\theta}_t)$$

- Two-scales algorithm.
- Can be adapted to continuous actions.
- Basis for SOTA algorithm.
- But close to on-line algorithm...

Outline

AlphaGo

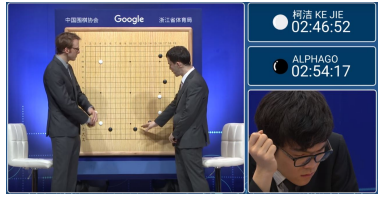
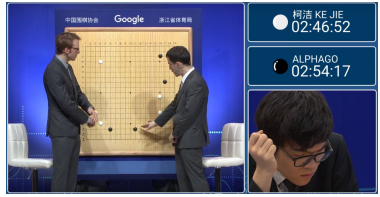


Figure 3 | MCTS in AlphaGo Zero. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus an upper confidence bound U that depends on a stored prior probability P and visit count N for that edge. b, The leaf node is expanded and the associated position s is evaluated by the neural network $(P(s, \cdot), V(s)) = f_\theta(s)$; the vector of P values are stored in the outgoing edges from s. c, Action value Q is updated to track the mean of all evaluations V in the subtree below that action. d, Once the search is complete, search probabilities π are returned, proportional to $N^{1/\tau}$, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.

AlphaGo

- Enhanced MCTS technique using a Deep NN for both the value function and the policy.
- Rollout policy and initial value network by supervised learning on a huge database.
- Enhancement of the value network using Actor/Critic RL on self-play.

AlphaGo



AlphaGo Zero

- No supervised initialization but only self-play.
- Alternate
  - MCTS with a current policy.
  - Gradient descent toward the resulting MCTS policy
- Much shorter training time and better performance!

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

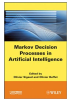
R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

References

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

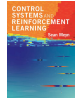


O. Sigaud and O. Buffet.
Markov Decision Processes in Artificial Intelligence.
Wiley, 2010

M. Puterman.
Markov Decision Processes. Discrete Stochastic Dynamic Programming.
Wiley, 2005

D. Bertsekas and J. Tsitsiklis.
Neuro-Dynamic Programming.
Athena Scientific, 1996

W. Powell.
Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions.
Wiley, 2022



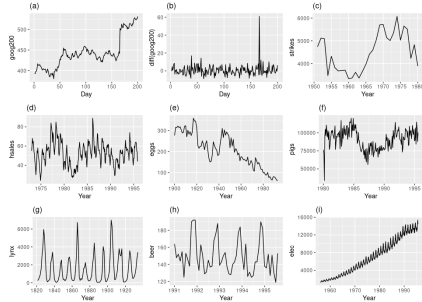
S. Meyn.
Control Systems and Reinforcement Learning.
Cambridge University Press, 2022

V. Borkar.
Stochastic Approximation: A Dynamical Systems Viewpoint.
Springer, 2008

T. Lattimore and Cs. Szepesvári.
Bandit Algorithms.
Cambridge University Press, 2020

Outline

Time Series

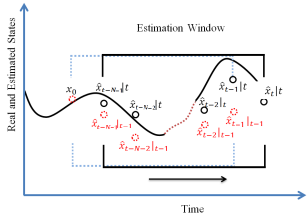


Time Series

- Sequence of values of the same entity across time.
- Values taken at regular interval, most of the time
- **Beware:** time dependency in the values!

Source: Hyndman and Athanasopoulos

Which Goals?



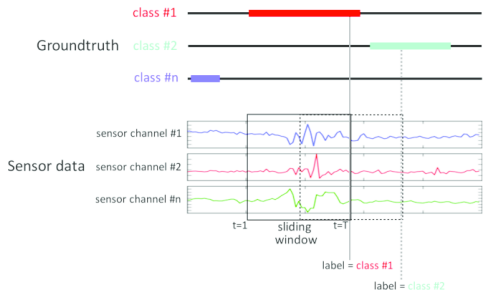
Goals

- Supervised:
  - Predict a value in the future,
  - Predict some values (a trajectory) in the future,
  - Predict a category in the future.
- Unsupervised:
  - Find break points,
  - Group some series together (possibly in real time)

- Using future values to act at a given time not allowed!

Source: N. Hashemian

Time Series and Structured Signals



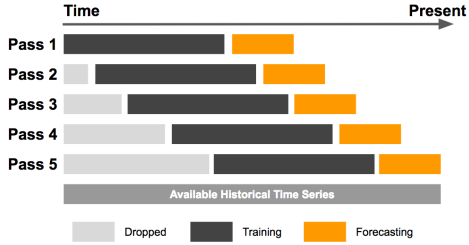
Structured Signals

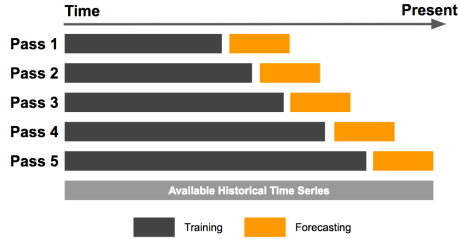
- Sequence of values of the same entity (spatially or temporaly).
- Decision can be taken a posteriori.
- No hard real-time constraints.

- Easier to deal with... but dependency with the data.

Source: D. Roggen

Time Series and Validation





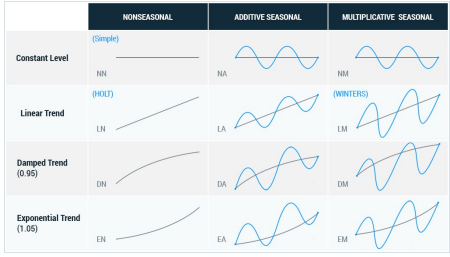
Cross Validation

- Never use the future. . . including for the validation.
- Classical Cross Validation is not working!
- Backtesting principle.

- Loss choice remains important.
- For structured data, safety buffer required between training and testing data.

Source: Uber

Trend and Seasonality



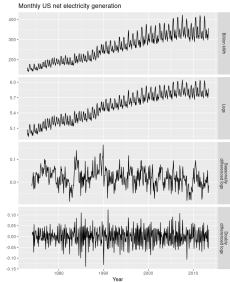
Trend and Seasonality

- Trend: long term evolution of average behavior.
- Seasonality: periodic variability around this mean.
- Residual: values after subtraction of the trend and the seasonality

- Need to estimate everything using only the past.

Source: E. Matveev

Stationarization



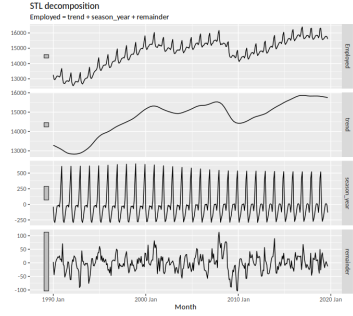
Stability in time assumption

- Required for learning...
- but not necessarily true.
- Often approximately correct after a transformation!

- Strongly data dependent!

Source: Hyndman and Athanasopoulos

Time Series Modeling



Models

- 3-layers approach: trend, seasonality and residuals.
- Decomposition not well specified...
- Several approaches for each layer!

Source: Hyndman and Athanasopoulos

Statistical Approach

$$\hat{X}_t \approx \sum_{j=1}^{p} \phi_j X_{t-j} + \sum_{k=1}^{q} \theta_k Z_{t-k} + \hat{Z}_t$$

Statistical Approach

- Most classical modeling.
- Combines past values of the sequence and a random noise.
- Explicit modeling of the variability!
- Complex estimation...

Source: H. Parra

Machine Learning Approach

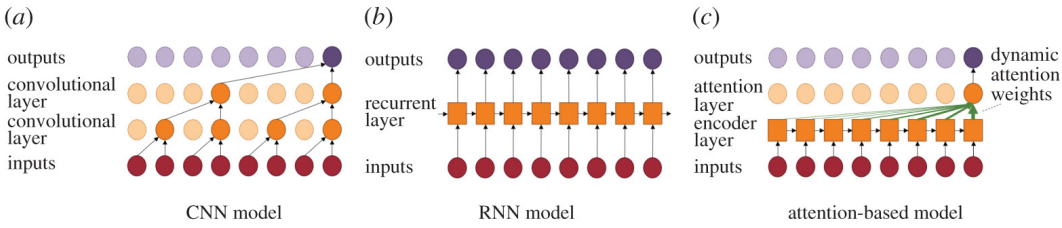
	Datetime	lag_1	lag_2	lag_3	lag_4	lag_5	lag_6	lag_7	Count
0	2012-08-25 00:00:00	NaN	NaN	NaN	NaN	NaN	NaN	NaN	8
1	2012-08-25 01:00:00	8.0	NaN	NaN	NaN	NaN	NaN	NaN	2
2	2012-08-25 02:00:00	2.0	8.0	NaN	NaN	NaN	NaN	NaN	6
3	2012-08-25 03:00:00	6.0	2.0	8.0	NaN	NaN	NaN	NaN	2
4	2012-08-25 04:00:00	2.0	6.0	2.0	8.0	NaN	NaN	NaN	2
5	2012-08-25 05:00:00	2.0	2.0	6.0	2.0	8.0	NaN	NaN	2
6	2012-08-25 06:00:00	2.0	2.0	2.0	6.0	2.0	8.0	NaN	2
7	2012-08-25 07:00:00	2.0	2.0	2.0	2.0	6.0	2.0	8.0	2
8	2012-08-25 08:00:00	2.0	2.0	2.0	2.0	2.0	6.0	2.0	6
9	2012-08-25 09:00:00	6.0	2.0	2.0	2.0	2.0	2.0	6.0	2

Machine Learning Approach

- Past taken into account only by feature engineering!
- Often using directly lagged values from the past.
- Variability not taken into account.
- Estimation with classical ML tools.

Source: A. Singh

Deep Learning Approach



Deep Learning Approach

- Past taken into account through the architecture.
- Explicit use of past values.
- Variability not taken into account.
- Huge choice for the architecture.

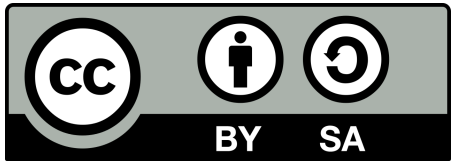
- Often trade-off performance/interpretability!

Source: B. Lim and S. Zohren

References

R. Hyndman and G. Athanopoulos.
Forecasting: principles and practice (3rd ed.)
OTexts, 2021

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

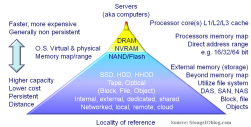
Outline

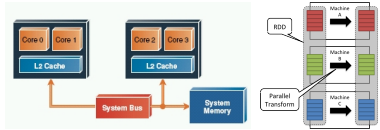
Too slow? Too big?

A frustrated Data Practicionner...

Source: Shutterstock

Big Data?





Hardware Constraints

- All the computations are done in a core using data stored somewhere nearby.
- Constraints:
  - Data access / storage (Locality of Reference).
  - Multiple core architecture (Parallelization).
  - Cluster (Distribution)

Sources: storageioblog.com / Unknown / R. Ho

What could be limiting?

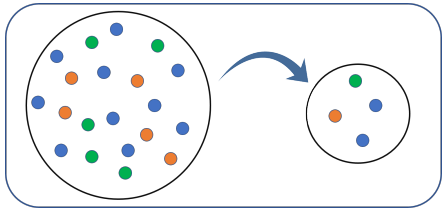
Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

Sampling Trick



- **Speed is linked to data size**
- Much faster with a smaller dataset!

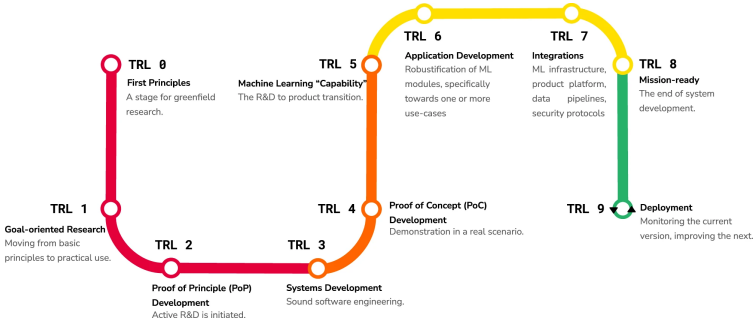
Data Sampling

- Similar idea than polling...
- Similar techniques to do it well (stratification!)

- Always a good idea when working with a large dataset...
- At least during a first exploration!
- **Rule of thumb:** Sample your data so that any experiment takes less than 5 minutes.

Source: S. Thatte

From POC to Production



From POC to Production

- POC: only first step(s)!
- Moving to production requires much more work: usability, scaling, IT integration...
- Main difficulty outside academia!

Source: Lavin et al.

POC: Proof of Concept

Outline

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

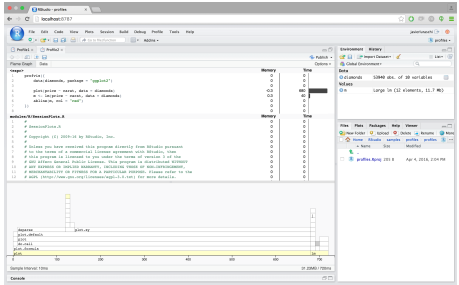
Possible Issues

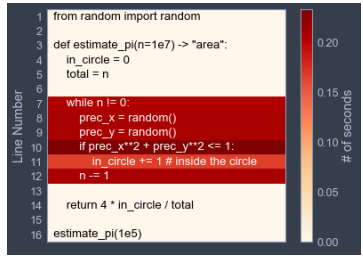
- **Coding issue?**
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- **Better algorithm/language/library? (code optimization)**
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What is slow?





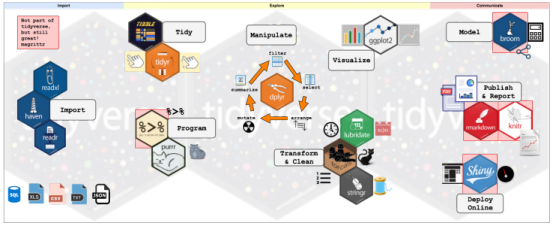
Profiling

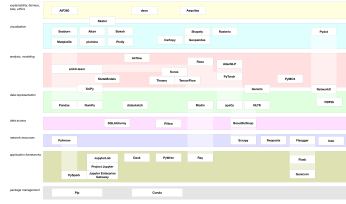
- Use a profiler to find out.
- Don't (over)optimize otherwise.
- Profiler in `RStudio`.
- Profiler in `Jupyter` (`line_profiler/py-heat-magick`), in another IDE or standalone (`yappi/py-spy/austin`).

- Think of using a debugger in case of incorrect results (and of making tests).

Source: RStudio/Serifovic

Libraries





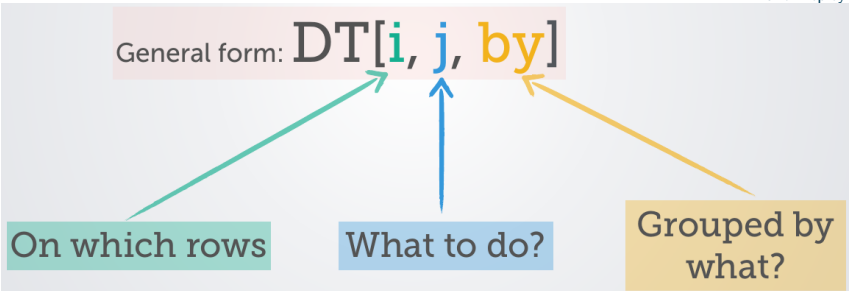
Libraries

- Avoid coding as much as possible...
- Pick a **good** implementation (often packaged in a library) based on:
  - capability,
  - product development,
  - community health.
- Choice may depend on goal/ecosystem!

- {tidyverse} is often a good starting point in R.

Sources: storybench.org/IBM

{data.table} - Datatable



Speed and memory optimized data.frame

- Complete rewrite of the R - Python structure.
- Standalone and optimized C code.
- Allow in place modification, grouping and fast indexing...

- {dplyr} is optimized for expressivity and connectivity.
- pandas is optimized for expressivity and speed.
- polars is another interesting option.

Source: r-datable.com

Algorithmic Design

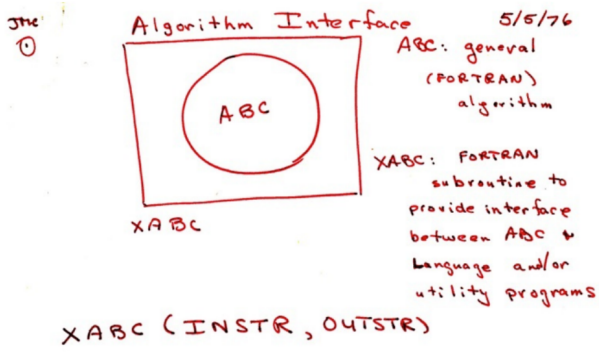
	Time					
Sort	Average	Best	Worst	Space	Stability	Remarks
Bubble sort	O(n^2)	O(n^2)	O(n^2)	Constant	Stable	Always use a modified bubble sort
Modified Bubble sort	O(n^2)	O(n)	O(n^2)	Constant	Stable	Stops after reaching a sorted array
Selection Sort	O(n^2)	O(n^2)	O(n^2)	Constant	Stable	Even a perfectly sorted input requires scanning the entire array
Insertion Sort	O(n^2)	O(n)	O(n^2)	Constant	Stable	In the best case (already sorted), every insert requires constant time
Heap Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Constant	Instable	By using input array as storage for the heap, it is possible to achieve constant space
Merge Sort	O(n*log(n))	O(n*log(n))	O(n*log(n))	Depends	Stable	On arrays, merge sort requires O(n) space; on linked lists, merge sort requires constant space
Quicksort	O(n*log(n))	O(n*log(n))	O(n^2)	Constant	Stable	Randomly picking a pivot value (or shuffling the array prior to sorting) can help avoid worst case scenarios such as a perfectly sorted array.

Complexity

- Algorithm choice can have a huge impact.
- Sorting algorithm example!
- Approximated/Stochastic variants...

Source: StackOverflow

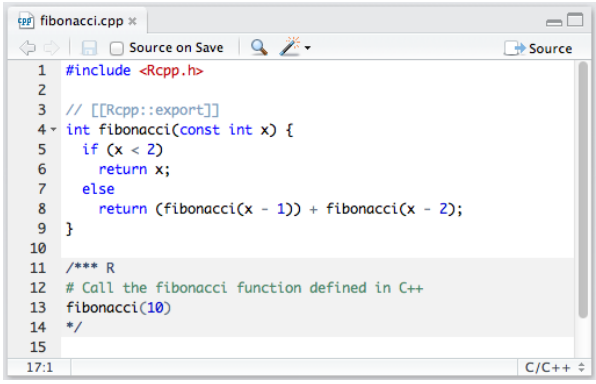
Faster Language



Interpreted vs Compiled

- `R` and `Python` are interpreted languages...
- constructed as a glue between libraries.
- Use compiled (and optimized) libraries... or compile code.

Source: J. Chambers



C++ in R

- Easy way to write functions in C++ and use them in R.
- Similar package to incorporate code from Python, Julia, Java, Scala...

Source: RStudio

Cython


```python
# python_functions.py
# ---------------------
import math

def f(x):
    return math.exp(-(x ** 2))

def integrate_f(a, b, N):
    s = 0
    dx = (b - a) / N
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

```python
# cython_functions.pyx
# ---------------------
from libc cimport math

cdef double f(double x):
    return math.exp(-(x ** 2))

def integrate_f(double a, double b, int N):
    cdef double s = 0
    cdef double dx = (b - a) / N
    cdef int i
    for i in range(N):
        s += f(a + i * dx)
    return s * dx
```

C/C++ from Python

- Easy way to write C/C++ code using a syntax a la Python
- Based on a static compiler.

- numba/jax are also interesting.

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

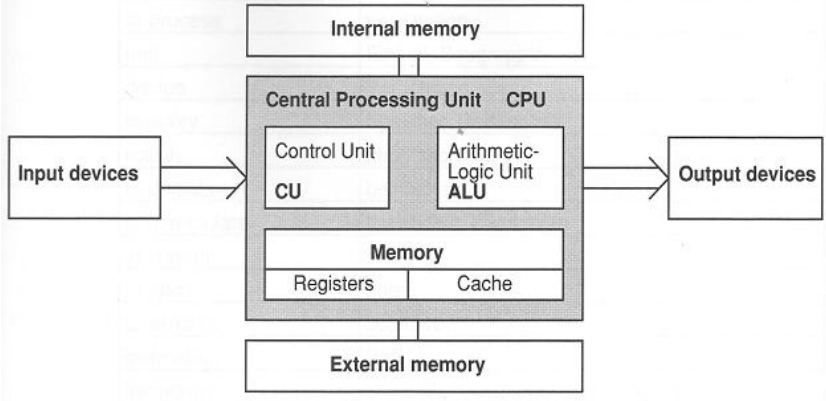
Possible Issues

- Coding issue?
- **I/O issue?**
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- **Better memory usage? (locality of reference)**
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

Computer Architecture

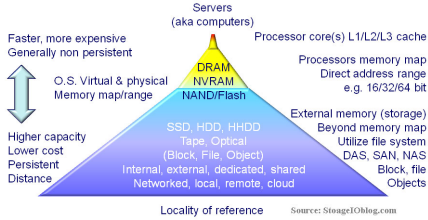


Central Processing Unit

- Everything should go through the CPU...

Source: Unknown

Memories



Size hierarchy

CPU register	64 b $\times$ 16
Level 1 cache access	32-65 kb per core
Level 2 cache access	256-512 kb per core
Level 3 cache access	8-32 MB shared
Main memory access	4 GB - 2 TB
Solid-state disk I/O	120 GB - 300 TB
Rotational disk I/O	250 GB - 20 TB

Source: storageioblog.com

Memories

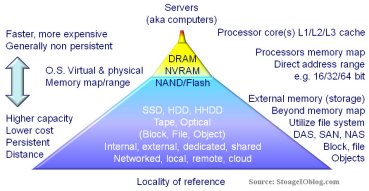
Speed hierarchy

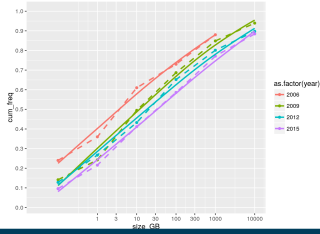
1 CPU cycle	0.3 ns	1 s	
Level 1 cache access	0.9 ns	3 s	
Level 2 cache access	2.8 ns	9 s	CPU bound latency
Level 3 cache access	12.9 ns	43 s	
Main memory access	120 ns	6 min	
Solid-state disk I/O	50 μs	2 days	
Local network	120 μs	3 days	
Rotational disk I/O	10 ms	12 months	IO bound latency
Internet: SF to NYC	40 ms	4 years	
Internet: SF to Australia	183 ms	19 years	
Read 1 MB sequentially from RAM	250 μs	10 days	
Read 1 MB sequentially from SSD disk	1 ms	40 days	IO bound bandwidth
Read 1 MB sequentially from HD disk	20 ms	2 years	

Source: SoftWare Yoga/P. Norvig

CPU: Central Processing Unit / I/O: Input/Output / OS: Operating System

Locality Of Reference



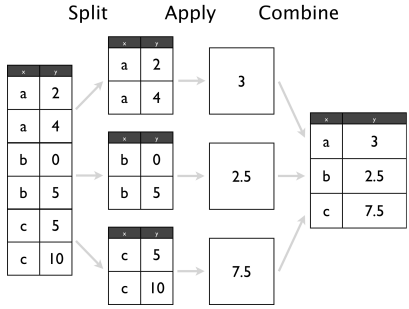


Memory Issue

- Data should be as **close** as possible from the core.
- **Ideal case:** dataset in the **memory of a single computer**.
- **Useless** if data used only once... (bottleneck = I/O)
- Memory required may be
  - **larger** than raw dataset (interactions...)
  - **smaller** than raw dataset (split...)
- Memory growth **faster** than data growth (fewer big data limitation in ML?)

Sources: storageioblog.com / Sz. Pafka

Split/Apply/Combine

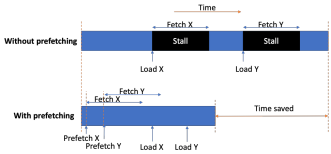


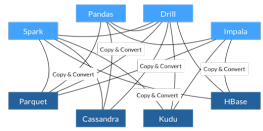
Split/Apply/Combine a.k.a. `GROUP BY`

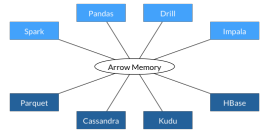
- Very simple strategy!
- Load in the memory only the data you need for the computation.
- Often much easier for production than for the learning part...

Source: H. Hofmann

I/O Optimization







Prefetching

- Pre-load data in background.

Zero Copy

- Avoid any copy/translation of data.
- Single representation of objects.
- `Apache Arrow` (combined with `Parquet`) is becoming a de facto standard.

Sources:: Arc Community/Arrow

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

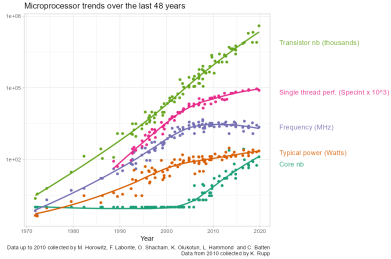
Possible Issues

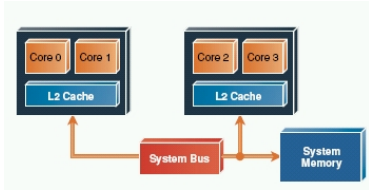
- Coding issue?
- I/O issue?
- **Processing issue?**
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- **Better CPU usage? (parallelization)**
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

Parallelization





Speed Issue

- **Parallelization**: Modern computer have **several cores**.
- **HPC / DS (HPDA) setting**: CPU bound tasks / I/O bound tasks.
- **Data science**: Often **embarrassingly parallel** setting (no interaction between tasks).
- Not always acceleration due to **I/O limitation**!

Sources: ELP / Unknown

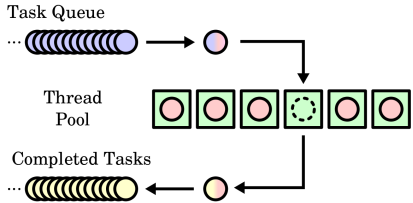
Parallel Computing with R

Embarassingly Parallel Algorithm

- Family of packages with a similar syntax to parallelize
  - the `apply` family,
  - the `do/dopar` loop.
- Different backends/implementations: thread/fork, MPI, client/slave...
- {`future`} proposes a high-level abstraction implementing a generic parallelization framework.

Source: E. McCallum & S. Weston

Parallelization in Python



Parallelization Tools

- Global Interpreter Lock makes thread less interesting for CPU bound tasks.
- `multiprocessing` library provides `Pool` and `Process` to parallelize tasks.
- `Pool` uses a map/apply approach with a fixed number of processes.
- Built-in in `Scikit-Learn` (`n_jobs` parameter) using `joblib`.
- Advanced functionalities (distribution/DAG) available in `Dask`/`Ray`

Source: Wikipedia

Outline

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- **Data storage issue?**

Enhancement?

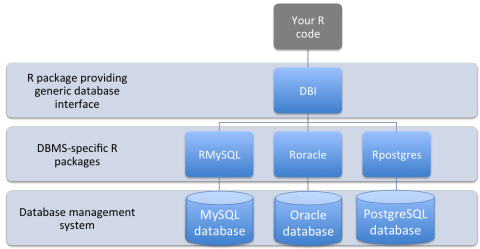
- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- **Better data storage? (database)**
- More computers? (distribution)
- Better computing infrastructure? (hardware)

Databases

(SQL?) Databases

- Most convenient tool to store/access data.
- Abstraction of the implementation that eases the use.
- Lot of knowledge inside.

Source: CodingGeek

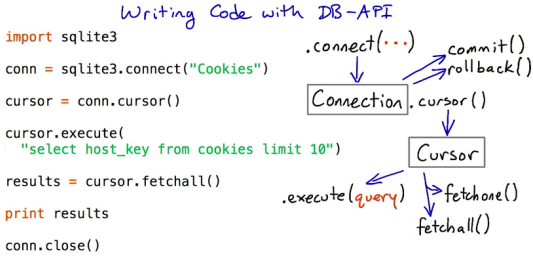


{DBI}, a DB API

- Standardized API for database.
- Several database specific packages.
- Connection with `dbConnect()`.
- Allow to send a request and retrieve the result `dbGetQuery()`.
- Can be used almost as easily as a local dataframe `tbl()` / `collect()` / `compute()`.

Source: R. Peng

DB API

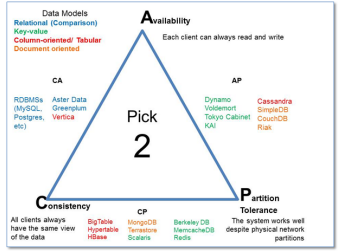


DB API

- Standardized API for database.
- Several database specific libraries...
- Allow to send a request and retrieve the result.
- `SQLAlchemy` allows to interact in a more pythonic way.

Source: Udacity

More than one solution: SQL/NoSQL



SQL

- Most classical design,
- Limitations linked to the CAP theorem: Hard to distribute without asking less...

NoSQL (Not only SQL!)

- Relaxation to ease distribution.
- Simplification/modification of the stored data type to ease the use.

Source: flux7.com

Why Not Always Use a (Meta) Database?

Unified (DB) interface

- Query (almost) any datastore from as single place.
- `Drill`/`Trino` supports a variety of relational databases, `NoSQL` databases and file systems.
- Both use `SQL`-like requests
- with `{sergeant}`/`{RPresto}`, `drill`/`Trino` can be used in `R`.
- with `py-drill`/`trino-python-client`, `drill`/`Trino` can be used in `Python`.

- `duckdb` is a lighter interesting option which supports local dataframe, local files and few databases including `duckdb` itself!

Source: drill.package.com

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

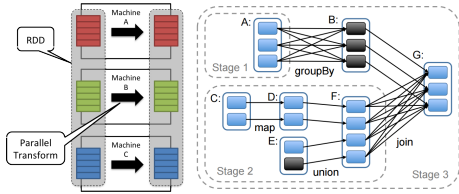
Possible Issues

- Coding issue?
- **I/O issue?**
- **Processing issue?**
- **Data storage issue?**

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- **More computers? (distribution)**
- Better computing infrastructure? (hardware)

Distribution

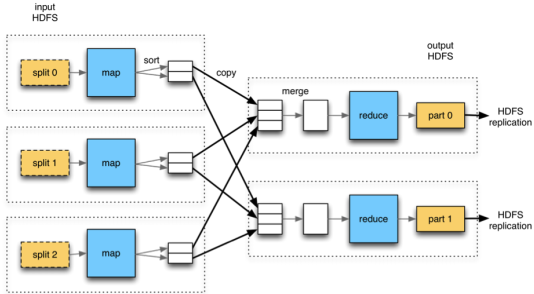


True Big Data Setting

- Computation in a **cluster**:
  - Distribution of the **data** (DS / HPDA),
  - or/and distribution of the **computation** (HPC)
- `Hadoop/Spark` realm.
- Locally **parallel in memory** computation are faster. . . if data used more than once.
- Real **challenge** when **not embarrassingly parallel** (interaction. . .)

Source: R. Ho / N. McBurnett

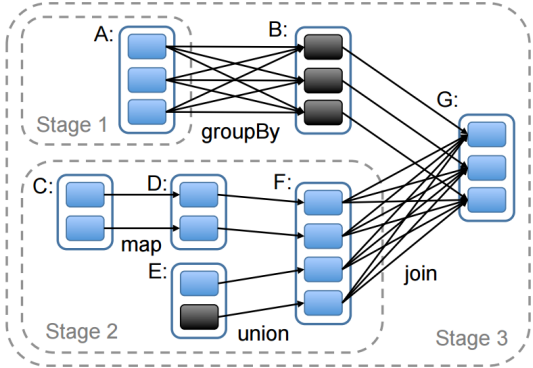
Hadoop and Map/Reduce



Hadoop

- Implementation of (classical) Map/Reduce algorithm.
- Data transfer through disk and networked file system!
- Main contribution: Node failure handling and ecosystem.

Source: Cloudera

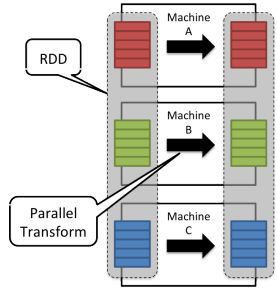


Spark

- More flexible algorithm structure (DAG).
- In Memory: cache some objects in memory...

Source: N. McBurnett

Distribution of UDF



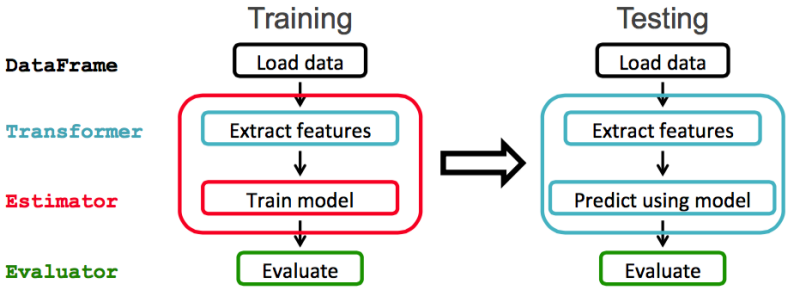
Spark as a a generic engine

- From single machine Spark usage to huge cluster.
- Dataframe API (/ RDD API)
- User Defined Function (UDF) can be applied.

Source: R. Ho

API: Application Programming Interface / RDD: Resilient Distributed Dataset

Distributed ML with Spark ML

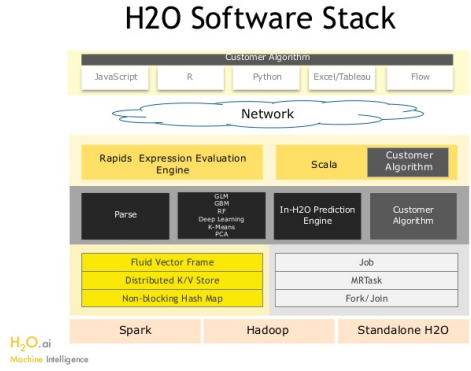


Spark

- Full distributed power of Spark
- ML Lib

Source: MapR

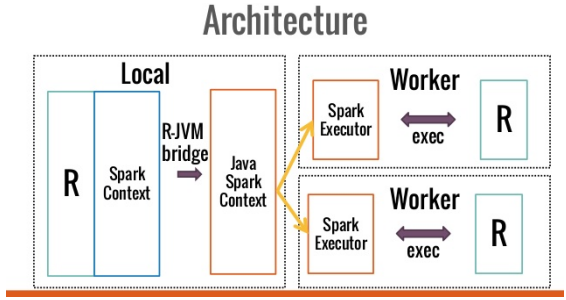
Distributed ML with H2O



Distributed ML system

- Standalone or `Spark` based
- Easy to use.

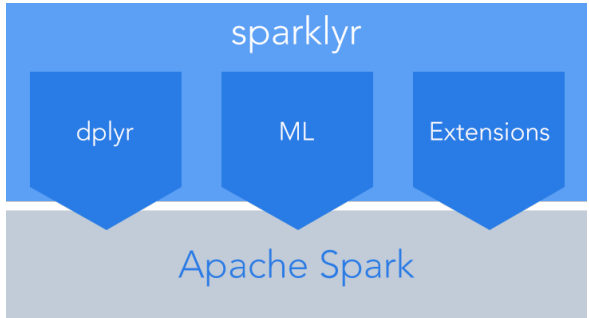
Source: h2o.ai



Official Spark R interface

- Allow working on DataFrame (data.frame like structure).
- Parallelized list apply with User Defined Functions available.

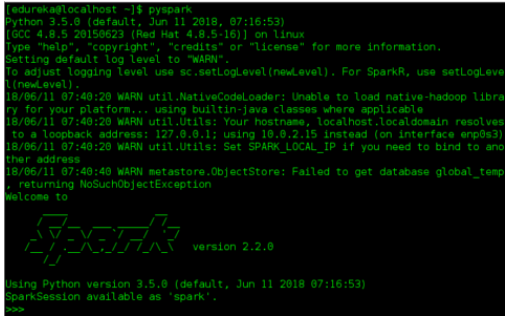
Source: S. Venkataraman & R. Sun



{Sparklyr}

- Convenient ML interface to `Spark` (or `h2o`).
- Convenient {`dplyr`} interface to `Spark`.
- Allow using more or less the same code as with {`dplyr`}.
- User Defined Functions also available.

Source: RStudio

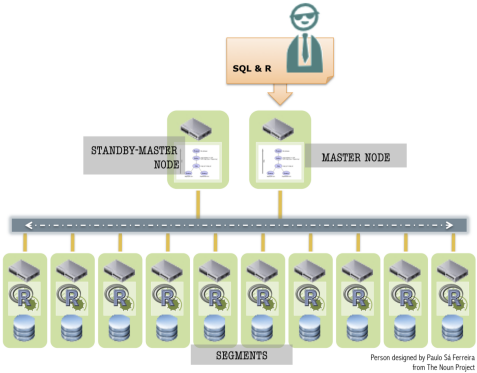


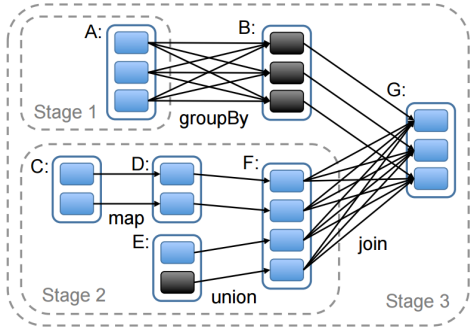
PySpark

- Provide access to both the DataFrame and RDD API.
- Access through `pyspark` rather than the usual `python` shell.
- User Defined Functions are available.

Source: K. Keshari

DB or Distributed System?





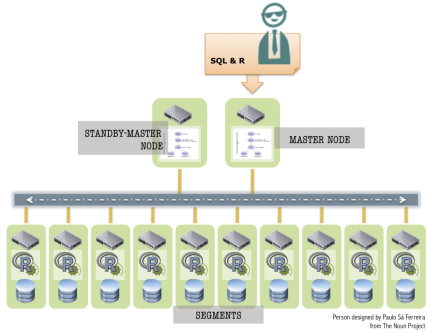
Database vs Distributed System

- DB: focus on data then computation.
- Distributed System: focus on computation then data.

- **Are they that different?**

Sources: Unknown / N. McBurnett

UDF: DB as a Distributed System

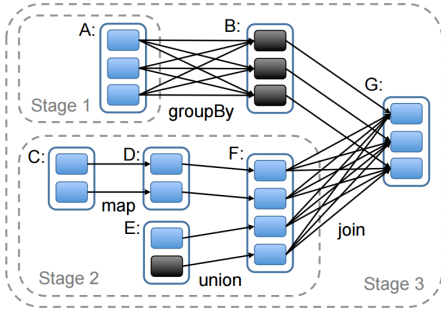


Database and User Defined Function

- Allow to defined complex function that can be run in the server of the DB.
- Idea: minimize the data transport by moving only the answer.
- `PostGreSQL`, `SqlServer`, `Oracle`, `Teradata`, `HAWQ`, `SAP Hana`...
- Require some priviledges...

Source: Unknown

SparkSQL: a Distributed System as a DB



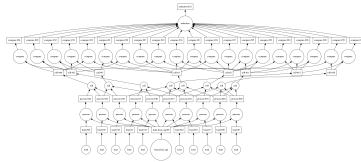
Spark SQL

Spark as a DB engine

- Store data files in disk/memory (caching).
- Use SparkSQL to request data from it.

Sources: N. McBurnett / spark.apache.org

Lighter Distribution Engines



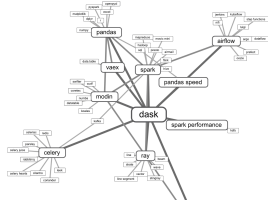
- `Hadoop`/`Spark` are often seen as complex to use...

Lighter Distribution Engines

- Based on the idea of chunking data and using a DAG to organize the computations.
- Several instantiations:
  - `dask`, `ray`, `vaex`, `PyArrow` in `Python`
  - {`future`}/{`targets`}, {`arrow`} in `R`
- Perform operations on dataset of arbitrary size using from 1 to 100 computers.
- Different implementation choices/maturities but promising direction.

Source: M. Rocklin

Dask / Ray / vaex / PyArrow ...

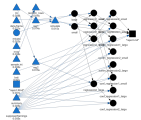


Dask / Ray / vaex / PyArrow ...

- Construct a task DAG on chunked data from a regular `Python` code (API à la `Pandas`/`NumPy`/`scikit-learn`).
- Execute this DAG on various parallel/distributed architecture.
- No connection with Spark ecosystem... but much more flexibility!
- Single computer out of core computations.

Source: M. Schmitt

{future}, {targets} and {arrow}



{Future} and promises

- Create {future} variable whose construction is not blocking until its further use..
- Abstraction used to implement a generic parallelization backend.

{targets}

- Build dependencies graph (a la make).
- Cache and parallelization!

{arrow}

- Chunked data.frame.

Source: ropensci.github.io/drake/

Outline

What could be limiting?

Possible Issues

- Coding issue?
- I/O issue?
- Processing issue?
- Data storage issue?

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- Better computing infrastructure? (hardware)

What could be limiting?

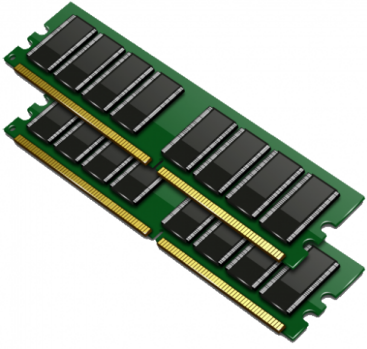
Possible Issues

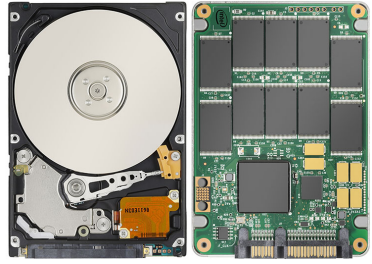
- Coding issue?
- **I/O issue?**
- **Processing issue?**
- **Data storage issue?**

Enhancement?

- Better algorithm/language/library? (code optimization)
- Better memory usage? (locality of reference)
- Better CPU usage? (parallelization)
- Better data storage? (database)
- More computers? (distribution)
- **Better computing infrastructure? (hardware)**

Memories





RAM and SSD

- The larger and the faster the better...
- Quite cheap nowadays.

Source: Unknown / Unknown

RAM: Random Access Memory / SSD: Solid-State Drive

Processing Units

PU: CPU, GPU, FPGA, ASICS

- More than one processor architecture.
- Flexibility vs performance.
- Parallelism: CPU $<$ GPU $<$ FPGA $<$ ASIC.

Cluster

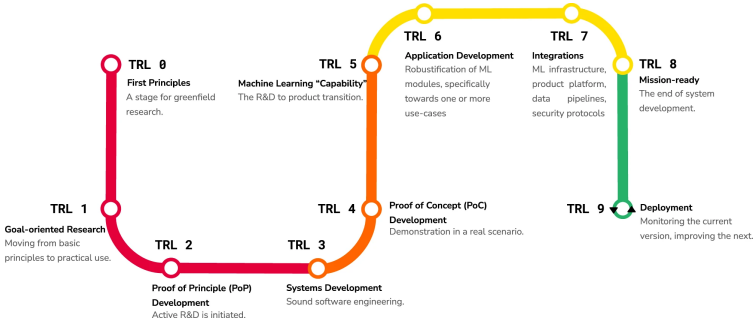
- More computers...
- I/O is important!

Source: Nvidia / advancedclustering.com

PU: Processing Unit / CPU: Central Processing Unit / GPU: Graphical Processing Unit / FPGA: Field Programmable Gate Array / ASIC: Application-Specific Integrated Circuit /

Outline

From POC to Production



From POC to Production

- POC: only first step(s)!
- Moving to production requires much more work: usability, scaling, IT integration...

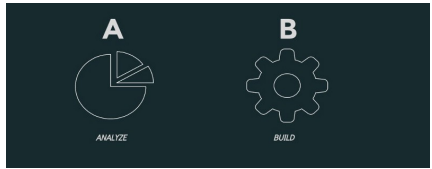
- Main difficulty outside academia!

Source: Lavin et al.

POC: Proof of Concept

Outline

Data Products



For Human - Insight (Study)

- Data / Analysis
- Most classical variations:
  - Report,
  - Static dashboard,
  - Interactive dashboard.

For Machine - Automation (Product)

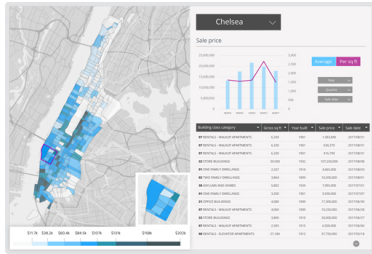
- Prediction / Modeling.
- Most classical variations:
  - Batch update,
  - On-demand

More Factors

- Data, Users, Temporal aspect, Location...

Source: E. Mandel

Insights



For Human - Insight

- Data / Analysis
- Most classical variations:
  - Report,
  - Static dashboard,
  - Interactive dashboard.

- No sophisticated algorithms are required to yield value!
- Huge data quality challenge!

Source: Appsilon

Insights

Report

- Analysis, AB testing, KPI...
- Word processor / Literate programming (Rmd/Notebook)

Static Dashboard

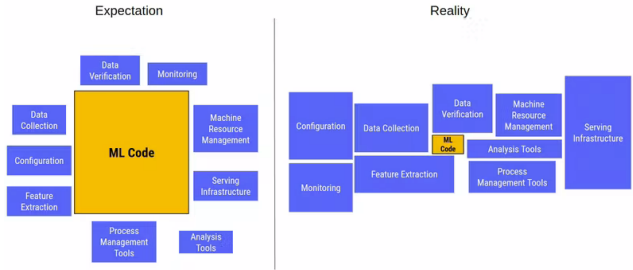
- Graph / Automatic summary...
- Literate programming (Rmd/Notebook) / Dataviz tools / Static web page

Interactive Dashboard

- Graph / Automatic summary with user interaction...
- Javascript / Client/server ({`Shiny`}/`Flask`/`Dash`)

Source: Appsilon

Automation



For Machine - Automation

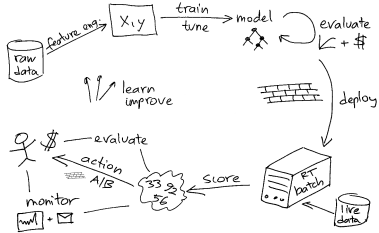
- Prediction / Modeling.
- Most classical variations: Batch update and On-demand

- Much more demanding!
- Going from POC to production is not easy.

Sources: import idea

POC: Proof Of Concept

Automation

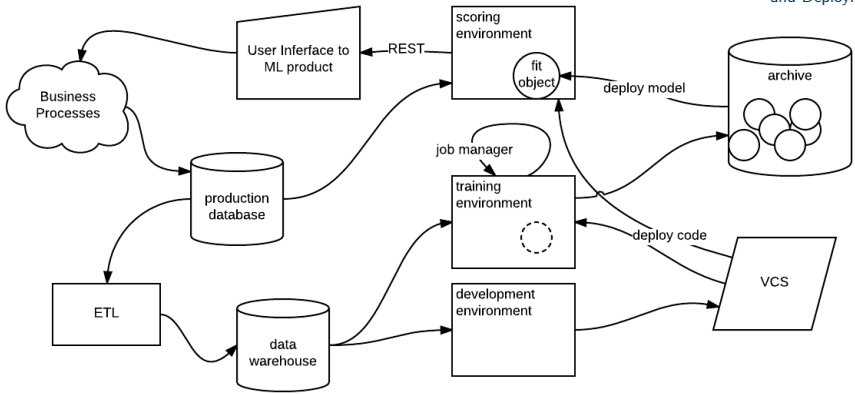


Using an algorithm in production

- Not the same hardware requirements for dev, training and prediction (CPU/RAM vs latency/availability/scalability).
- Better to use the same language/code everywhere.
- Often require data (cleaning) duplication.
- Two quite different scenarios:
  - Batch scoring (easier)
  - On-demand (REST API, Stream...)

Source: Sz. Pafka

DS Architecture



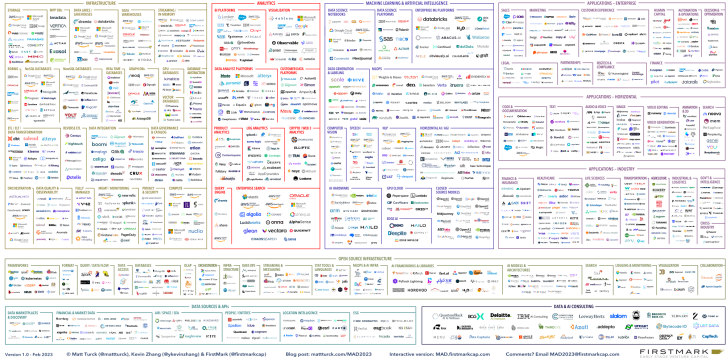
Data Science Architecture

- Usage dependent architecture!
- **Finding a good architecture is difficult**

Sources: Sz. Pafka

Outline

More tools



Much more tools!

- Much more tools than analytics, database and distribution!
- BI/Dataviz, Prediction delivery, DS platform, Data Pipeline, Orchestration...

BI/DataViz

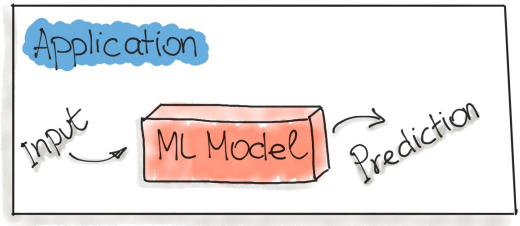
DataViz

- BI/Dataviz dedicated tools.
- Specific development with R and Python (Niche?).

- Quite mature ecosystem...

Source: OSDC

Prediction Delivery



How to deliver the predictions?

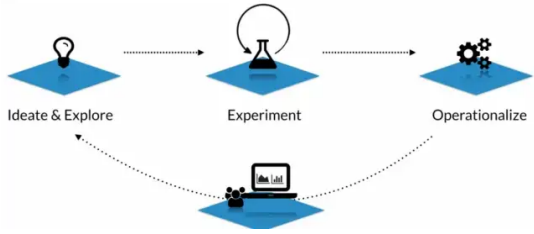
- By running the code...
- By delivering the code.
- By delivering the model (`PMML`/`PFA`) ?
- By delivering an API

- Should not be done manually?

Source: ML-Ops.org

PMML: Portable Model Markup Language/PFA: Portable Format for Analytics

Data Science Platform



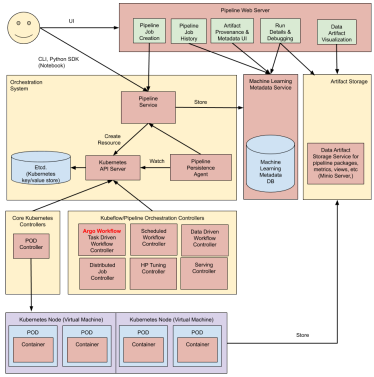
Data Science Platform

- Development and deployment.
- Code / low code / No code.
- Library / Style choices.

- Key to efficient delivery!

Source: OSDC

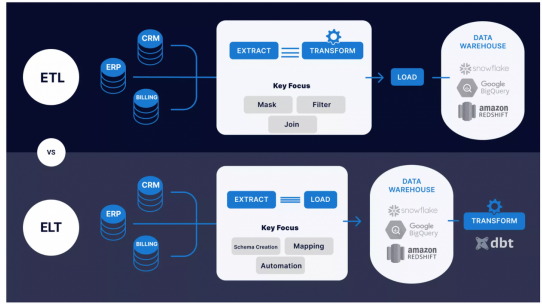
Orchestration



Orchestration

- Training/Predicting/Monitoring.
- Stream.
- Hardware/Software optimization.

Data Pipeline



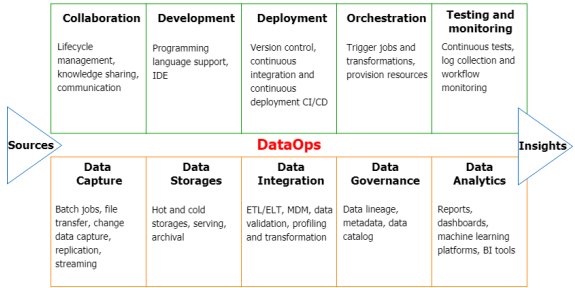
Data Pipeline

- Data preparation.
- Scaling issues.

- Data Management aspect!

Outline

DataOps/MLOps Approach

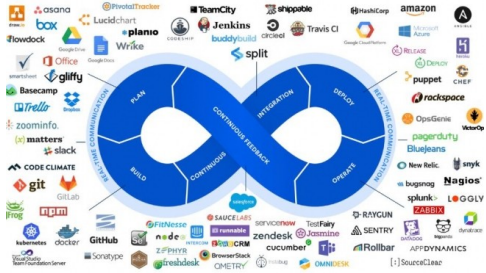


DataOps/MLOps

- Inspired by DevOps and Lean Management
- Mindset + tools to deal with Data products

Source: V. Maksimavicius

DevOps?

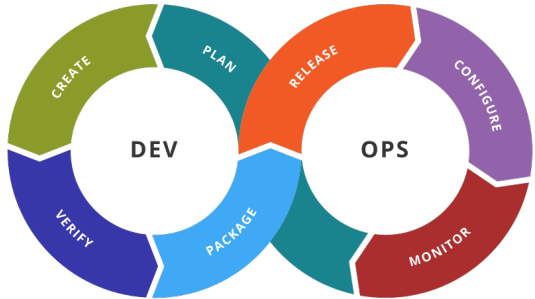


DevOps

- Combination of Software Development and IT Operations.
- *a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality*
- Combine tools and mindset!

Source: Atlassian?

DevOps Mindset

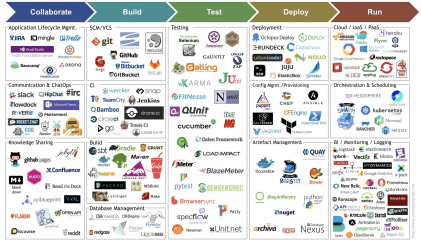


Much more than technical tools!

- **Culture:** Cooperation / Learning / Blamelessness / Empowerment
- **Automation:** Tools / Tests / Package / Configuration
- **Monitoring:** Dashboard / Post Mortem
- **Sharing:** Goals / Practice / Learning

Source: Wikipedia

DevOps Tools



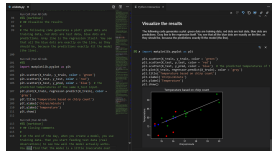
Lots of tools for each step!

- **Collaborate:** Lifecycle mgmt, Communication, Knowledge sharing
- **Build:** SCM/VCS, CI, Build, DB mgmt
- **Test:** Testing
- **Deploy:** Deployment, Config mgmt, Artifact mgmt
- **Run:** Cloud/*aas, Orchestration, Monitoring

- Tool choice depends on the context.
- Good usage is more important that the tool itself.

Source: James Bowman

Code and DevOps



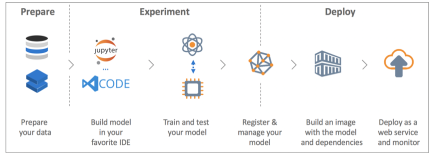
- Code are meant to be used/shared/reused.

Good practice

- Versioning (Code),
- Documentation,
- Testing,
- Packaging,
- Continuous Integration/Continuous Deployment,
- Human Training

Source: Microsoft

Models and MLOps



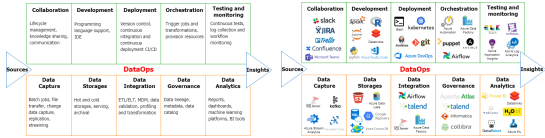
- Models are meant to be used/shared/reused.

Good practice

- Versioning (Models/Code/Dataset),
- Artifact mgmt,
- Documentation,
- Training/Testing/Monitoring,
- Human Training,
- Continuous Integration/Continuous Deployment

Source: Azure

Data and DataOps



- Data are meant to be used/shared/reused.

Good practice

- Versioning (Data/Processing),
- Documentation/Governance,
- Testing/Monitoring,
- Packaging (Feature store),
- Human Training,
- Continuous Integration/Continuous Deployment.

Source: Valdas Maksimavicuis

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
***Data Management at Scale*.**
O'Reilly, 2020

References

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

G. Harrison.
Next Generation Databases: NoSQL, NewSQL, and Big Data.
Apress, 2015

B. Chambers and M. Zaharia.
Spark, The Definitive Guide.
O'Reilly, 2018

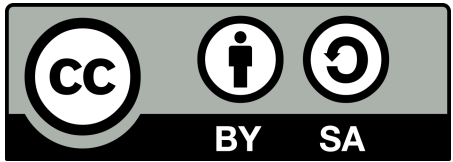
J. Davis and K. Daniels.
Effective DevOps.
O'Reilly, 2016

J. Chambers.
Extending R.
CRC Press, 2016

H. Wickham.
Advanced R (2nd ed.)
CRC Press, 2019

M. Gorelick and O. Ozsvald.
High Performance Python.
O'Reilly, 2020

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.

Outline

References

T. Hastie, R. Tibshirani, and J. Friedman.
The Elements of Statistical Learning.
Springer Series in Statistics, 2009

M. Mohri, A. Rostamizadeh, and A. Talwalkar.
Foundations of Machine Learning.
MIT Press, 2012

A. Géron.
Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow (3rd ed.)
O'Reilly, 2022

Ch. Giraud.
Introduction to High-Dimensional Statistics.
CRC Press, 2014

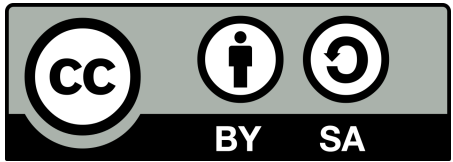
K. Falk.
Practical Recommender Systems.
Manning, 2019

R. Sutton and A. Barto.
Reinforcement Learning, an Introduction (2nd ed.)
MIT Press, 2018

T. Malaska and J. Seidman.
Foundations for Architecting Data Solutions.
O'Reilly, 2018

P. Strengholt.
Data Management at Scale.
O'Reilly, 2020

Licence and Contributors



Contributors

- Main contributor: E. Le Pennec
- Contributors: S. Boucheron, A. Dieuleveut, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Matzner, M. Sangnier, E. Scornet.