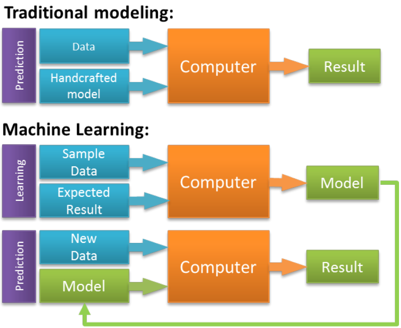
Machine Learning
Statistical Learning:
Introduction and Cross Validation

Ana Karina Fermin

winter 2020-2021

Outline

Machine Learning



A definition by Tom Mitchell (http://www.cs.cmu.edu/~tom/)

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: azavea

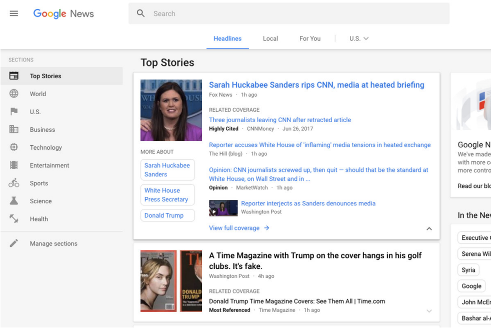
Credit Score, Bank Risk, ...

Credit Score

- **Task:** Prediction (default or no default)
- **Data:** Client profile, Client credit history...
- **Performance measure:** error rate.

Source: A. Fermin

News Clustering

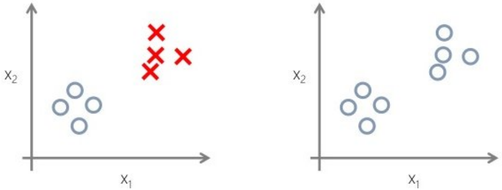


A news clustering algorithm:

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

Source: theverge.com

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

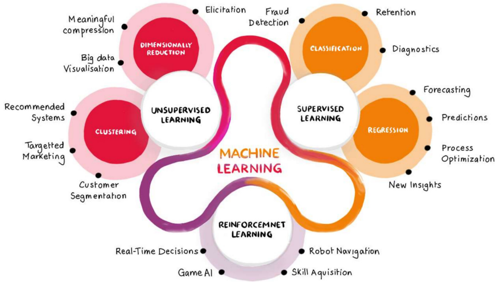
A Robot that Learns

A robot endowed with a set of sensors playing football:

- **Task**: play football
- **Performance**: score evolution
- **Experience**:
  - current environment and outcome,
  - past games

Source: Rutgers Prep School

Three Kinds of Learning



Unsupervised Learning

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

Supervised Learning

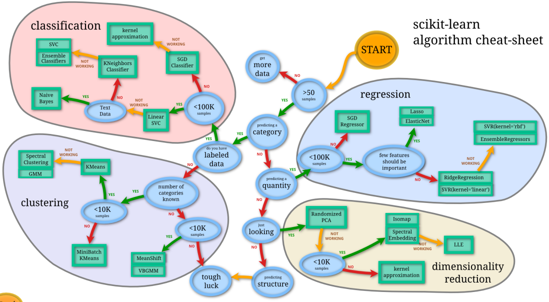
- **Task:** Prediction
- **Performance:** Average error
- **Experience:** Predictions (Ground Truth)

Reinforcement Learning

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

Source: BCG

Machine Learning (Unsupervised/Supervised)



Back

ML Methods

- Huge catalog of methods,
- Need to define the performance, feature design.
- **Here, we will only see supervised learning**

Source: scikit-learn.org

Credit Score, Bank Risk, ...

Credit Score

- **Task:** Prediction (default or no default)
- **Data:** Client profile, Client credit history...
- **Performance measure:** error rate.

Source: A. Fermin

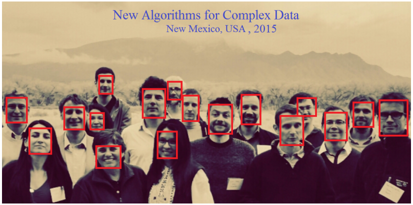
Spam detection (Text classification)

Spam detection

- **Task:** Prediction (spam or no spam)
- **Data:** email collection
- **Performance measure:** error rate.

Source: http://www.jesperdeleuran.dk/

Detection

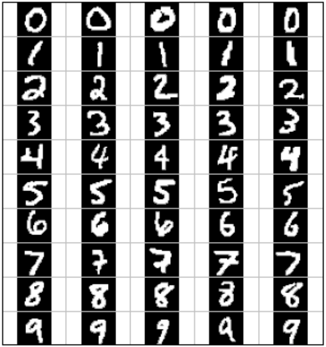


Face detection

- **Task:** Detect the position of faces in an image
- Different setting?
- Reformulation as a supervised learning problem.
- **Goal:** Detect the presence of faces at several positions and scales.
- **Data:** $\underline{X}$ = 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

Number

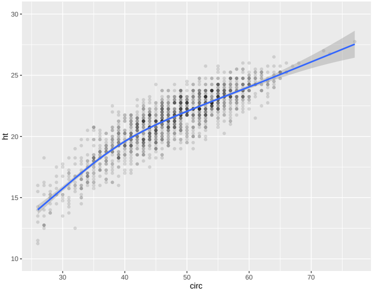


Reading a ZIP code on an envelop

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

Source: Y. LeCun

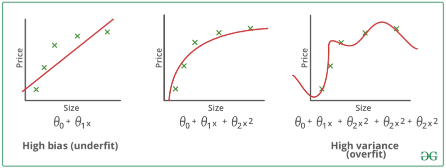
Eucalyptus

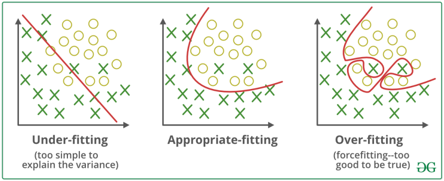


Height estimation

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

Under and Over Fitting



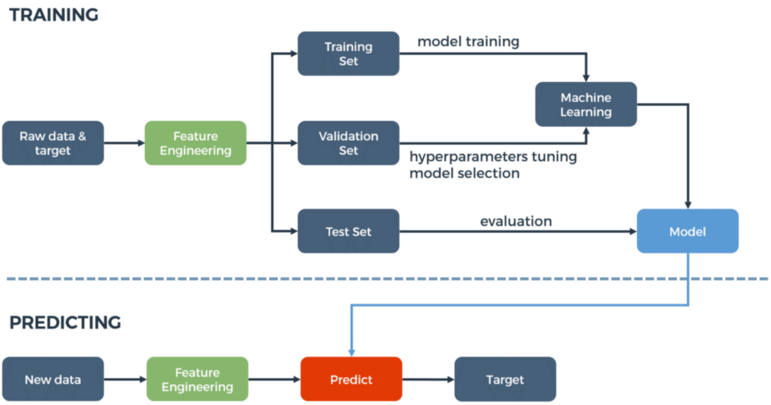


Finding the Right Complexity

- 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

ML Pipeline



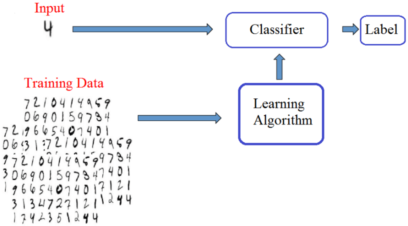
Source: CDiscount

Learning pipeline

- Test and compare models.

- Deployment pipeline is different!

IFMA/IMP Bloc 3 ML - Goal



Goal

- Know the inner mechanism of the most classical supervised **ML methods** (Logistic, SVM, Neural Nets and Trees) in order to understand their strengths and limitations.
- Understand some **optimization** tools used in ML

Evaluation

- A project (homework) with R

Schedule

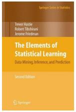
IFMA/IMP Bloc 3 ML - 5 Lectures (09h00-12h15)

- Fri. 15/01: Statistical Learning: Introduction and Cross Validation
- Tue. 19/01: ML Methods: Probabilistic Point of View
- Tue. 26/01: ML Methods: Optimization Point of View
- Tue. 02/02: ML Methods: SVM
- Tue. 09/02: ML Methods: Trees and Ensemble Methods, Neural Networks, etc

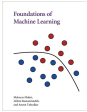
- 12/03: **Homework (project with 2-3 students)**

- 12/03: **Homework (project with 2-3 students)**
- For this homework :
  - You will have to build a good predictor from a dataset that I will give you.
  - The goal is not necessarily to obtain the best performance but to perform the work of a good data scientist.
  - You need to describe the task and the dataset using descriptive statistics and graphics.
  - You should explain how you have obtained your best predictor both in term of strategy and error estimation.
  - You are expected to describe the strength and the limitation of your approach and to propose some possible enhancements.
  - You may use R or Python. If you a use a notebook, please provide a compiled version.
  - The report should consists of around 20-30 pages and is much more than a code listing.
  - Originality of the work will be taken into account and any plagiarism will be sanctioned.

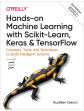
References



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 (2nd ed.)
O'Reilly, 2019

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:
  - Prediction 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:

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

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

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

Best Solution

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

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

Bayes Predictor (explicit solution)

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

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

- In regression with the quadratic loss

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

Issue: Solution requires to **know** $\mathbb{E}\left[Y|\underline{X}\right]$ 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

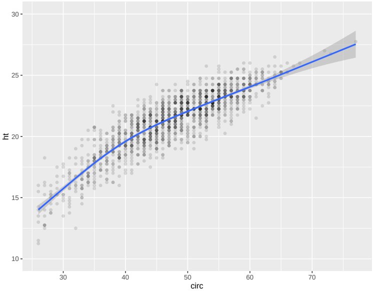
- One restricts f to a subset of functions $\mathcal{S} = \{f_\theta, \theta \in \Theta\}$
- One replaces the minimization of the average loss by the minimization of the empirical loss

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

- Examples:
  - Linear regression
  - Linear discrimination with

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

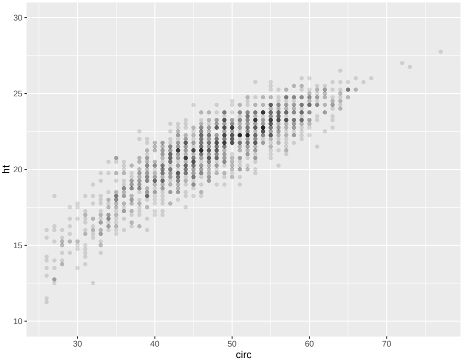
Eucalyptus



Height estimation

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

Eucalyptus

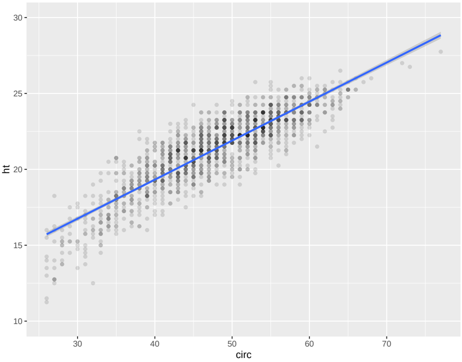


Dataset - P.A. Cornillon

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

- Can we predict the height from the circumference?

Eucalyptus

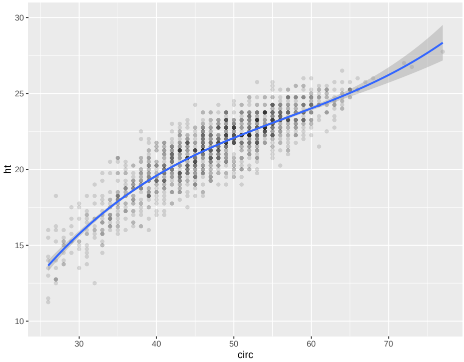


Dataset - P.A. Cornillon

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

- Can we predict the height from the circumference?
  - by a **line**?

Eucalyptus

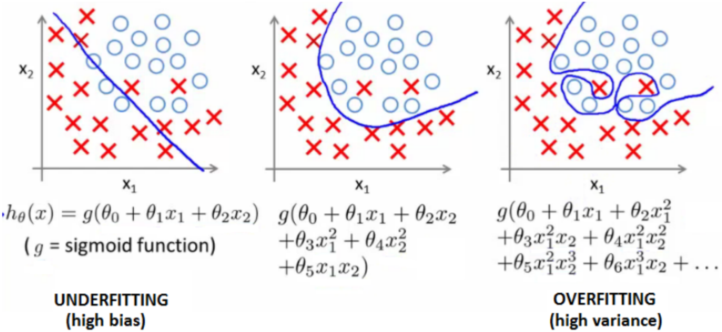


Dataset - P.A. Cornillon

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

- Can we predict the height from the circumference?
  - by a **line**? by a more complex formula?

Under-fitting / Over-fitting Issue

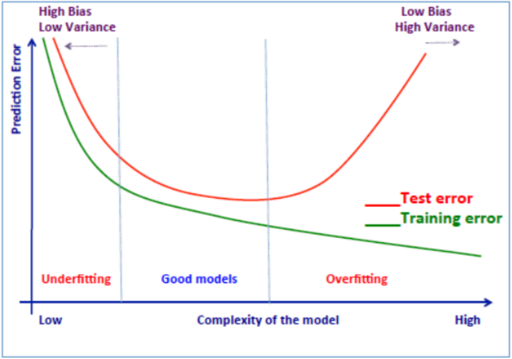


Model Complexity Dilemna

- 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



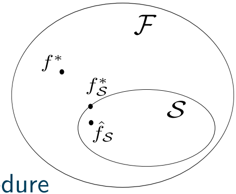
Under-fitting / Over-fitting

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

Source: Unknown

Bias-Variance Dilemma

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



Approximation error and estimation error (Bias/Variance)

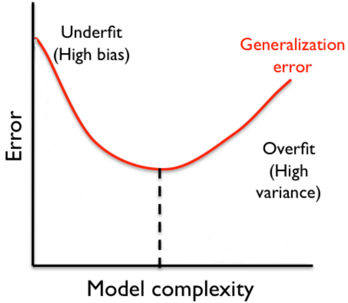
$$\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^*) = \underbrace{\mathcal{R}(f^*_{\mathcal{S}}) - \mathcal{R}(f^*)}_{\text{Approximation error}} + \underbrace{\mathcal{R}(\widehat{f}_{\mathcal{S}}) - \mathcal{R}(f^*_{\mathcal{S}})}_{\text{Estimation error}}$$

- 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

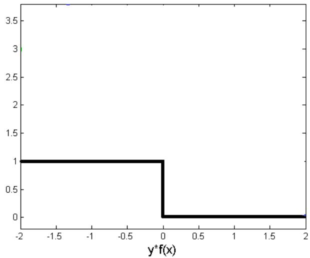
Theoretical Analysis

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

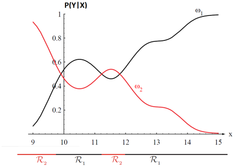


Empirical Risk Minimizer

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

- 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^* (which is independent of $\mathcal{D}_n$) is

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

Bayes Predictor (explicit solution)

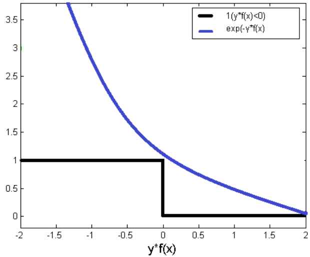
In binary classification with $0-1$ loss:

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

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

Source: A. Fermin

Optimization Point of View
Loss Convexification



Minimizer of the risk

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

- **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

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

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

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 ℓ' and minimize the empirical loss: **SVR, SVM, Neural Network, Tree, Boosting...**

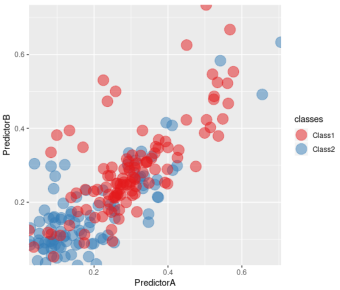
Outline

Example: TwoClass Dataset

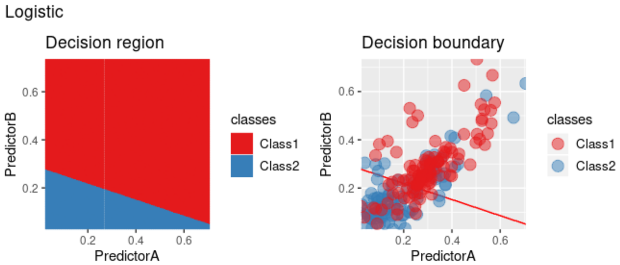
Synthetic Dataset

- Two features/covariates.
- Two classes.

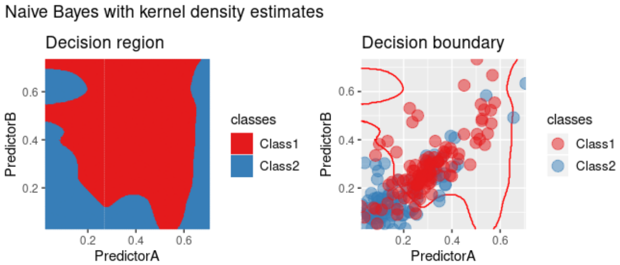
- Dataset from *Applied Predictive Modeling*, M. Kuhn and K. Johnson, Springer
- Numerical experiments with `R` and the `caret` package.



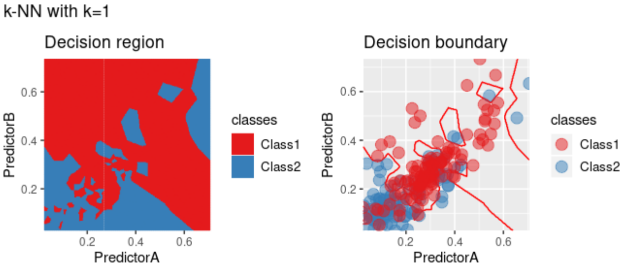
Example: Linear Discrimination



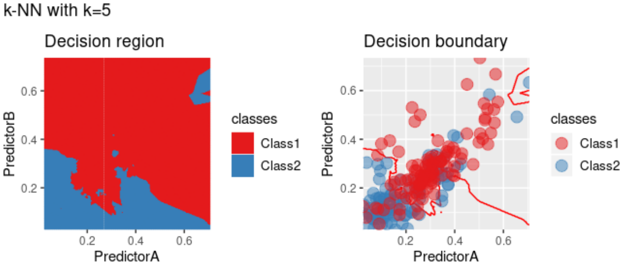
Example: More Complex Model



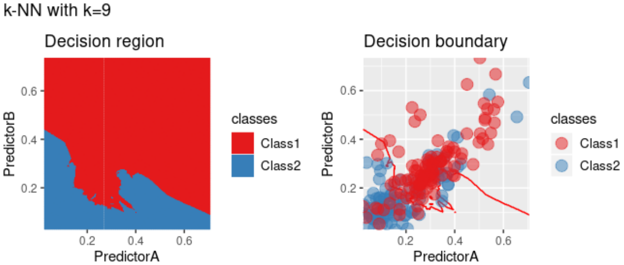
Example: KNN


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

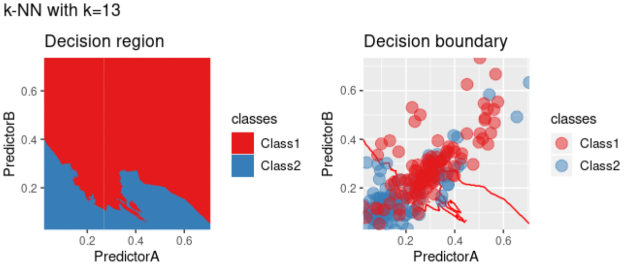
Example: KNN


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

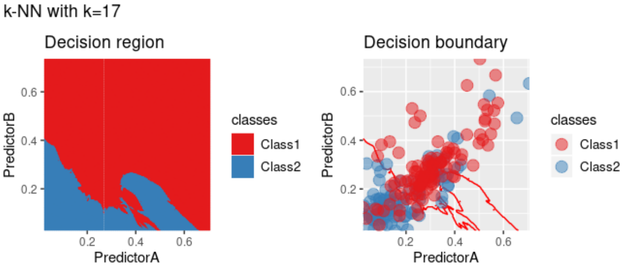
Example: KNN


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

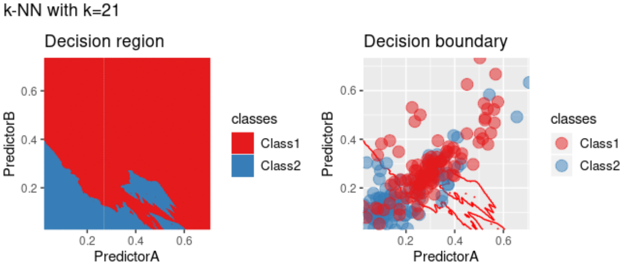
Example: KNN


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

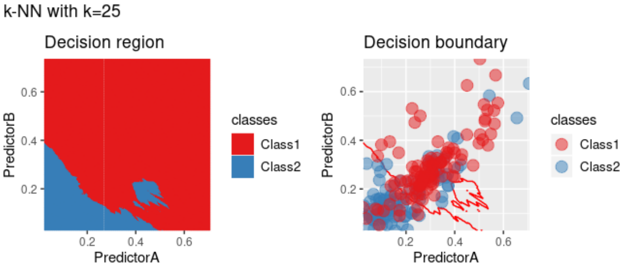
Example: KNN


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

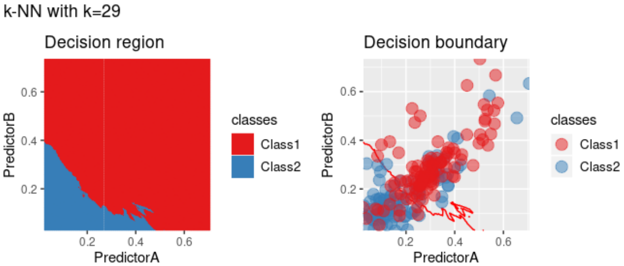
Example: KNN


k-NN with k=21
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

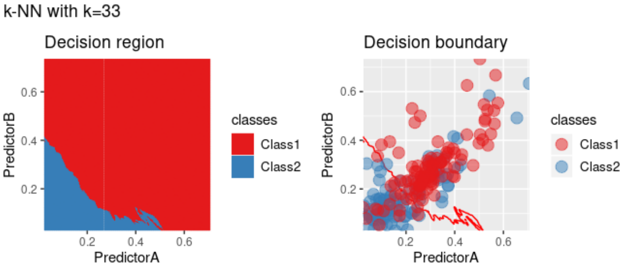
Example: KNN


k-NN with k=25
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

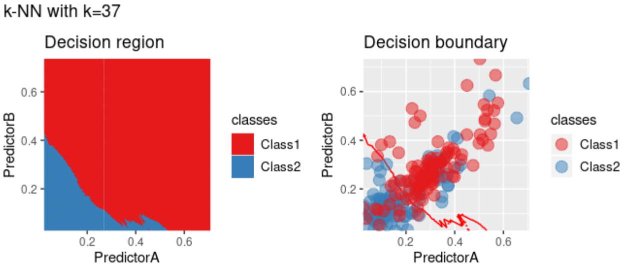
Example: KNN


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

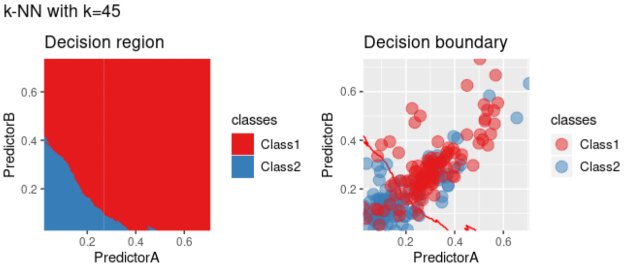
Example: KNN


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

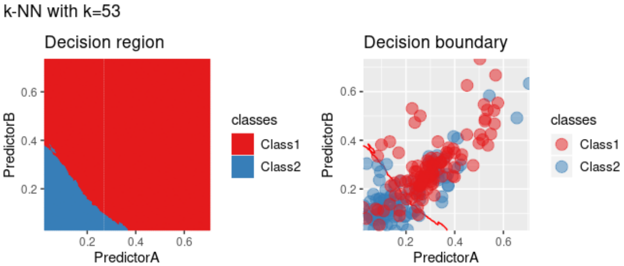
Example: KNN


k-NN with k=37
Decision region
PredictorB
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

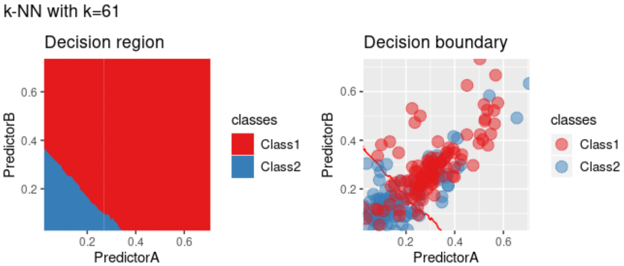
Example: KNN


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

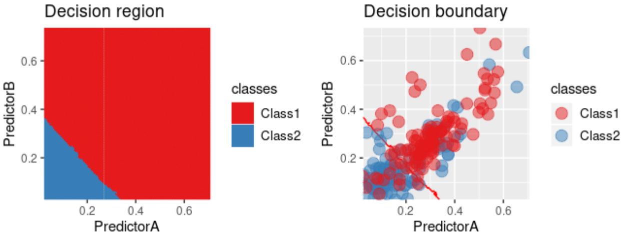

k-NN with k=53
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
classes
Class1
Class2

Example: KNN

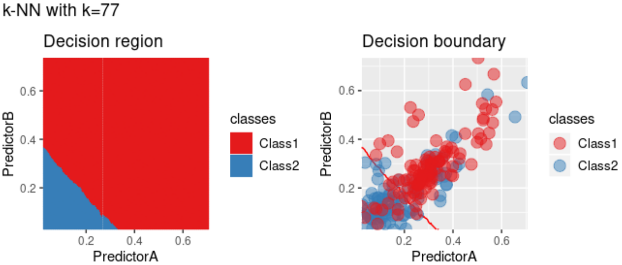

k-NN with k=61
Decision region
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: KNN

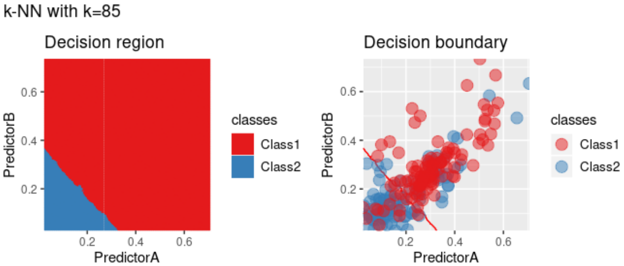
k-NN with k=69



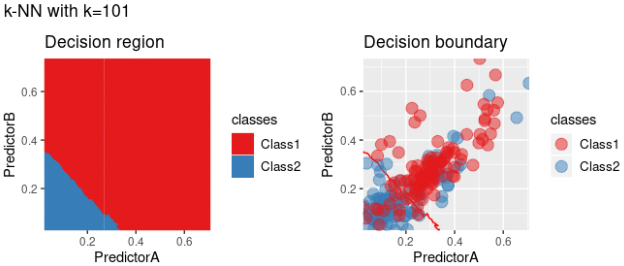
Example: KNN


k-NN with k=77
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

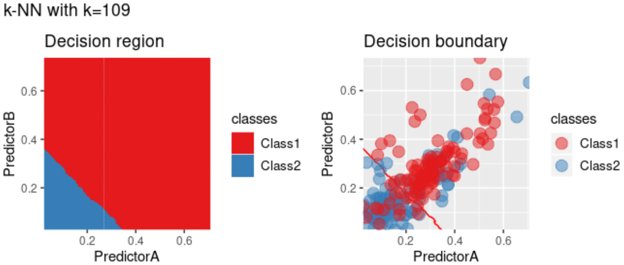
Example: KNN


k-NN with k=85
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: KNN

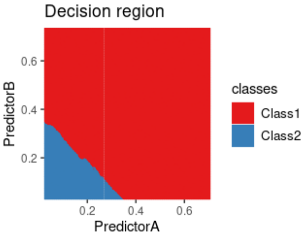

k-NN with k=101
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
classes
Class1
Class2

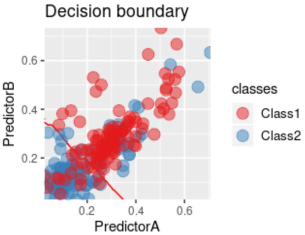
Example: KNN


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

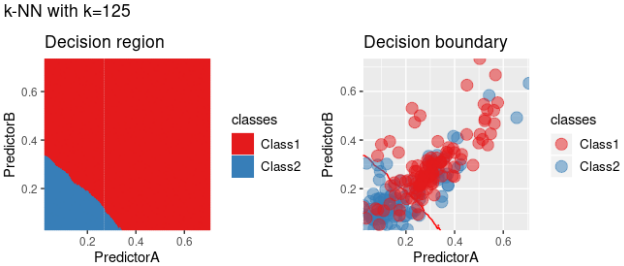
Example: KNN

k-NN with k=117

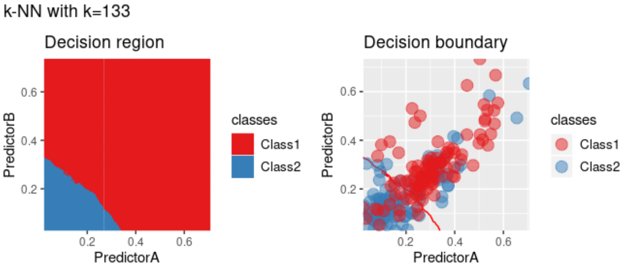




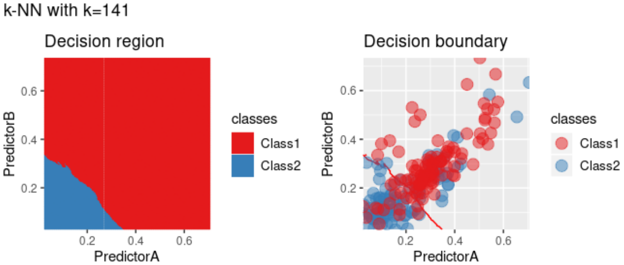
Example: KNN


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

Example: KNN

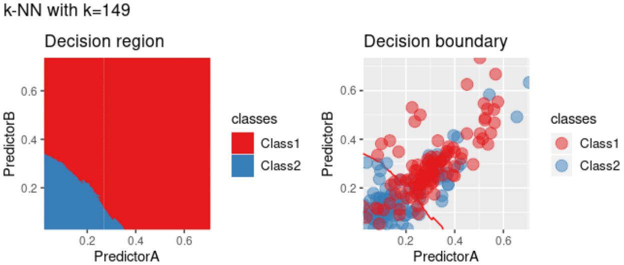

k-NN with k=133
Decision region
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2
Decision boundary
PredictorB
PredictorA
0.2
0.4
0.6
classes
Class1
Class2

Example: KNN

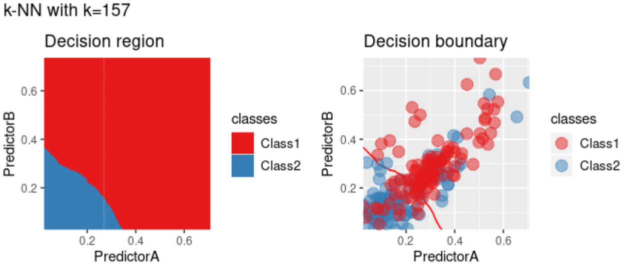


k-NN with k=141
Decision region
Decision boundary
PredictorA
PredictorB
0.2
0.4
0.6
classes
Class1
Class2

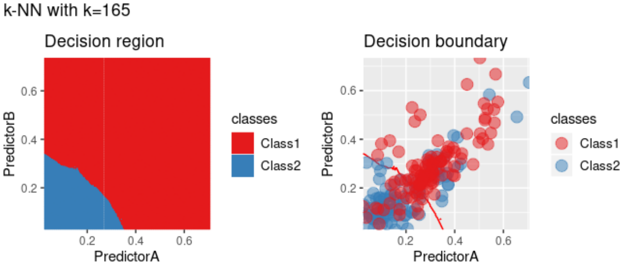
Example: KNN


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

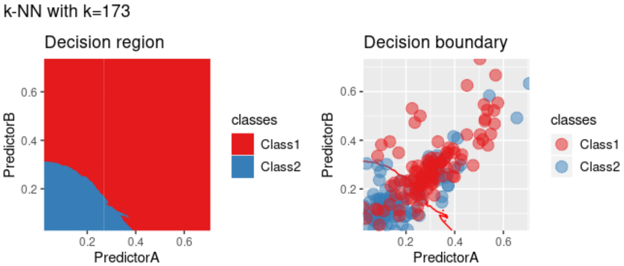
Example: KNN


k-NN with k=157
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

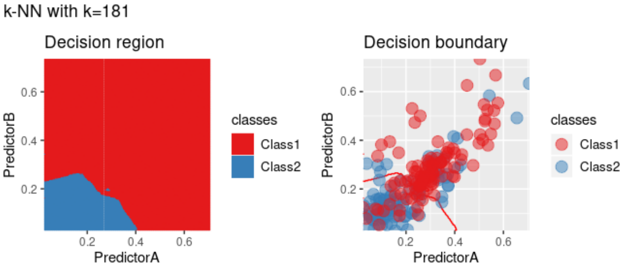
Example: KNN


k-NN with k=165
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

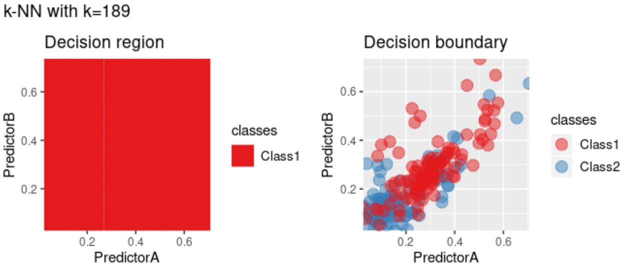
Example: KNN


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

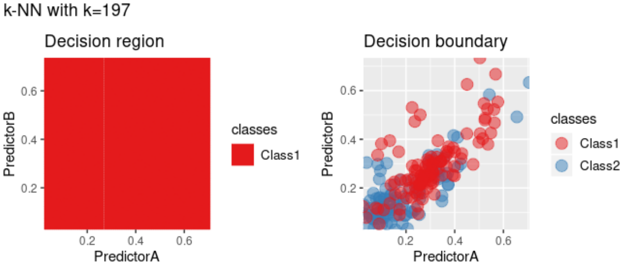
Example: KNN


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

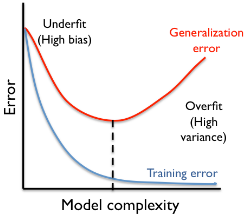
Example: KNN


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
0.2
0.4
0.6
PredictorA
classes
Class1
Decision boundary
classes
Class1
Class2

Training Error Issue



Error behaviour

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

- 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 error!

Source: JMP

Error Estimation vs Method Selection

Predictor Error Estimation

- **Goal:** Given a predictor f assess its quality.
- **Method:** Hold-out error computation (/ Error correction).
- **Usage:** Compute an estimate of the error 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 (/ Error correction)
- **Usage:** Compute error 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 Error Correction

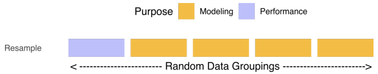
Two Approaches

- **Cross validation:** Very efficient (and almost always used in practice!) but slightly biased as it target uses only a fraction of the data.
- **Correction approach:** use empirical loss criterion but *correct* it with a term increasing with the complexity of $\mathcal{S}$
$$R_n(\widehat{f_{\mathcal{S}}}) \to R_n(\widehat{f_{\mathcal{S}}}) + \mathrm{cor}(\mathcal{S})$$
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!

Cross Validation



- **Very simple idea:** use a second learning/verification set to compute a verification error.
- Sufficient to remove the dependency issue!
- Implicit random design setting...

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 error 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 error 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 Error Estimation

- Use $\hat{f}^{HO}$ as predictor.
- Use $\mathcal{R}_n^{HO}(\widehat{f}^{HO})$ as an estimate of the error 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 error,
- 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 error on the subset $\mathcal{D}_{\text{test}}$:

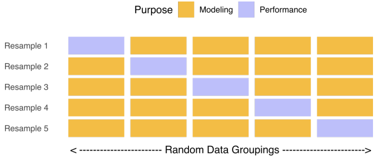
$$\mathcal{R}_n^{HO}(\widehat{f}^{HO}) = \frac{1}{n\epsilon} \sum_{(\underline{X}_i, Y_i) \in \mathcal{D}_{\text{test}}} \ell(Y_i, \widehat{f}^{HO}(\underline{X}_i))$$

- Only possible setting for error 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 error:

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

- Compute the average empirical error:

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

- 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:

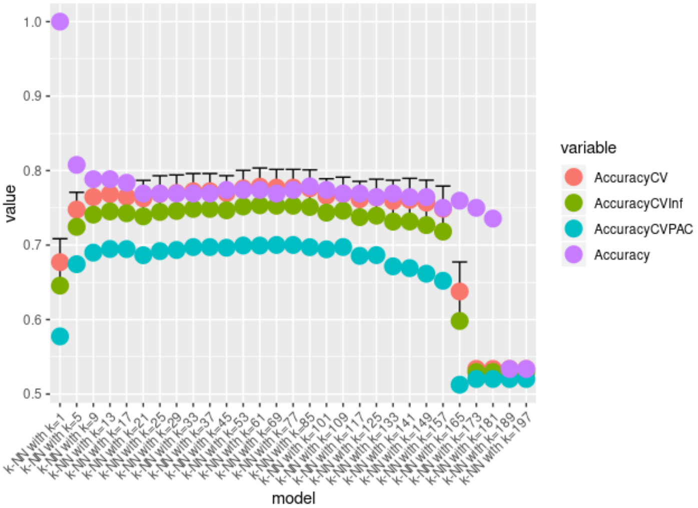
$$\mathbb{E}\left[\mathcal{R}_n^{CV}(\widehat{f})\right] = \mathbb{E}\left[\mathcal{R}_n^{-v}(\widehat{f}^{-v})\right]$$

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

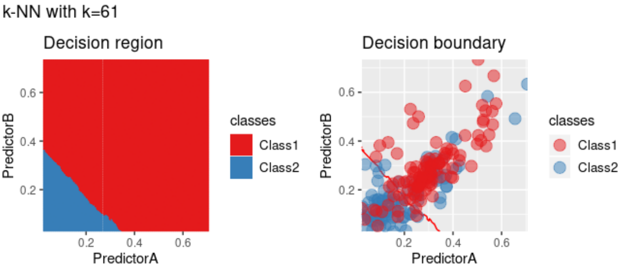
- Average risk for a sample of size $(1-\frac{1}{V})n$.
- Variance term much more complex to analyze!
- Fine analysis shows that the larger V the better...

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

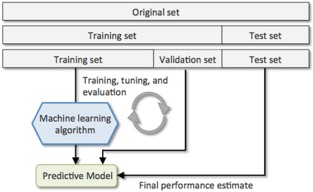
Cross Validation



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



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 error 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`.

Error Correction

- Empirical loss of an estimator computed on the dataset used to chose is 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

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

where $\text{pen}(\theta)$ is an error correction (penalty).

Penalties

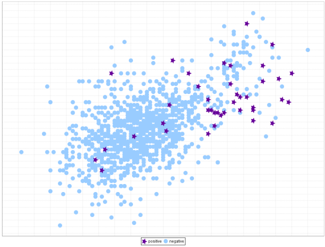
- Upper bound of the optimism of the empirical loss
- Depends on the loss and the framework!

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

Unbalanced and Rebalanced Dataset



Unbalanced Class

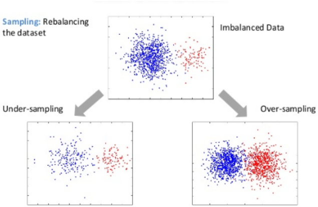
- **Setting:** One of the class is much more present than the other.
- **Issue:** Classifier *too attracted* by the majority class!

Rebalanced Dataset

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

Source: University of Granada

Resampling Strategies



Resampling

- Modify the training dataset so that the classes are more balanced.
- **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 error target:

$$\mathbb{E}_{\pi_t}\left[\ell(Y, f(\underline{X}))\right] = \sum_k \pi_t(k)\mathbb{E}\left[\ell(Y, f(\underline{X}))|Y = k\right]$$

Training

- Training class prob.: $\pi_{tr}(k)$
- Training Error target:

$$\mathbb{E}_{\pi_{tr}}\left[\ell(Y, f(\underline{X}))\right] = \sum_k \pi_{tr}(k)\mathbb{E}\left[\ell(Y, f(\underline{X}))|Y = k\right]$$

Implicit Testing Error Using the Training One

- Amounts to use a weighted loss:

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

- 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 error target:
$$\mathbb{E}\left[C(Y)\ell(Y, f(\underline{X}))\right]$$

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

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

- The Bayes classifier is now:

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

Bayes Predictor

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

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

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

- Allow to put more emphasis on some classes than others.

Linking Weights and Proportions

Cost and Proportions

- Testing error target:

$$\mathbb{E}_{\pi_t}\left[C_t(Y)\ell(Y, f(\underline{X}))\right] = \sum_k \pi_t(k) C_t(k) \mathbb{E}\left[\ell(Y, f(\underline{X})) | Y = k\right]$$

- Training error target

$$\mathbb{E}_{\pi_{tr}}\left[C_{tr}(Y)\ell(Y, f(\underline{X}))\right] = \sum_k \pi_{tr}(k) C_{tr}(k) \mathbb{E}\left[\ell(Y, f(\underline{X})) | Y = k\right]$$

- **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 reduced the size of a dataset without loosing a low probability class!

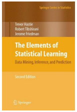
Combining Weights and Resampling

- **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 $C_{tr}(k) = C_t(k)\pi_t(k)/\pi_{tr}(k)$

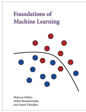
- Most ML methods allow such weights!

Outline

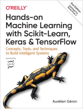
References



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 (2nd ed.)
O'Reilly, 2019

Licence and Contributors

Contributors

- S. Boucheron, A.K. Fermin, S. Gadat, S. Gaiffas, A. Guilloux, Ch. Keribin, E. Le Pennec, E. Matzner, E. Scornet and X Exed team.