Machine learning

Département GMM
Mastère Spécialisé VALDOM
Année 2022-2023

INSA de Toulouse

Philippe BESSE - Béatrice LAURENT -Cathy MAUGIS - Olivier ROUSTANT

Contents

Chapter 1

Introduction

Summary

The aim of this course is to introduce the supervised learning techniques most commonly used in *data science* for *decision-making aid* in many fields of application: industrial applications, marketing, insurance, biology, medicine ... The main objective is to built a model for forecasting and therefore to search for optimal models for different classical statistical algorithms (linear or generalized linear models, discriminant analysis), less classical (penalized regression, binary decision trees) or even so-called learning algorithms (random forests, neural networks, support vector machines, aggregation models) from *machine learning*.

1 Statistical learning/ Machine learning

Statistical learning and Machine learning play a key role in many fields of sciences, medicine, industry, marketing, insurance ..

As soon as a phenomenon is too complex or even too noisy to access an analytical description leading to a deterministic modeling, a set of approaches have been developed in order describe and model it from a series of observations. Let us see the historical steps of the development of statistical learning, machine learning, data science and artificial intelligence.

1.1 From Statistic to Artificial Intelligence through Data Science

1930-70 h-Octets Statistical inference

1950 Beginnings of Artificial Intelligence: Allan Turing

1970s kO Data analysis and *exploratory data analysis*

1980s MO Neural networks, functional data analysis

1990s GO *Data mining*: pre-acquired data

2000s TO Bioinformatics: $p >> n$, *Machine Learning*

2008 Data Science

2010s PO Big Data p and n very large

2012 *Deep Learning*

2016 Artificial Intelligence (IA): AlphaGo, Imagenet, Generative Adversarial Networks ..

VVV... : Volume, Variety, Velocity...

The development of data storage and computing resources gives rise to the production and the storage of a huge amount of data from which the *data scientist* will try to learn crucial informations to better understand the underlying phenomena or to provide predictions. Many fields are impacted, here are some examples of learning problems:

- **Medicine:** identify the risk factors for a certain type of cancer, based on clinical and demographic variables
- **Meteorology:** predict an air pollution rate based on weather conditions
- **Energy:** forecast an electricity consumption curve for a customer as a function of climatic variables and specific characteristics of this customer, build a model for energy optimization of buildings, or predict the energy production of a wind farm.
- **Consumers preferences data:** Websites and supermarkets collect a huge amount of data on the behavior of consumers. Machine learning algorithms are used to valorize these data (gathered sometimes with personal data such as age, sex, job, address ..) for recommandation systems, fixing personalized prices ..
- **Risk modeling:** construct a substitution model for a complex numerical code which allows to predict a map of the concentration of a pollutant in a soil after an accidental release. The objective is to perform a sensitivity analysis on the numerical code.
- **Genomics:** DNA microarrays allow to measure the expression of thousands of genes simultaneously on a single individual. It is, for example, a challenge to try to infer from those kind of data which genes are involved in a certain type of cancer, by comparing expression levels between healthy and sick patients. This is generally a high dimensional problem: number p of genes measured on a microarray is generally much larger than the number n of individuals in the study.
- **Aeronautical engineering:** Aerospace industry produces a huge amont of signal measurements obtained from thousand of on-board sensors. It is particularly important to detect possible anomalies before launching the satellite. Similarly, many sensors are involved in planes and it is important to detect a abnormal behavior on a sensor. The main objectives are curve clustering or classification and anomaly detections in a set of curves for predictive maintenance purposes.
- **Images:** Convolutional neural networks and deep learning led to impressive progresses for image classification. Many fields are concerned: medical images (e.g. tumor detection), earth observation satellite images, computer vision, autonomous vehicles, ...
- **Geolocalisation data:** Machine learning based on geolocalisation data has also many potential applications: targeted advertising, road traffic forecasting, monotoring the behavior of fishing vessels ...

The main reference for this course is the book " The elements of Statistical Learning" by T. Hastie et al [21]. Studying a certain phenomenon (presence

of a cancer or not/ abnormal behavior or not/ interest for a certain product ..), it is a challenge to derive which explanatory variables (among a possibly large number of available ones) are influent for the phenomenon of interest, as well as to provide a prediction rule. The main objective is therefore a *modeling* objective which can be specified into sub-objectives that have to be clearly defined prior the study since this determines the methods that can be implemented:

Explore, represent, describe, the variables, their correlations ..

Explain or test the influence of a variable or a factor in a specific model, assumed to be *a priori* known

Predict & Select a (small) set of predictors, to obtain an interpretable model, for example searching for biomarkers

Predict by a "black box" without the need for an explicit interpretation.

Important parameters of the problem are the dimensions: the number n of observations or sample size and the number p of variables observed on this sample. The high dimensional framework, where p is possibly greater than n has received a great interest in the statistical literature these last 20 years and specific methods have been developed for this non classical setting. We will see the importance of *parcimony*: "it is necessary to determine a model that provides an adequate representation of the data, with as few parameters as possible".

Historically, statistical methods have been developed around this type of problems and one has proposed *models* incorporating on the one hand *explanatory or predictive variables* and, on the other hand, a random component or *noise*. It is then a matter of *estimating* the *parameters* of the model from the observations, *testing* the significance of a parameter, *selecting* a model or a small set of influent variables, using the vocabulary of **statistical learning**. The main methods are implemented in the software R.

In the same time, the IT community talks more about **machine learning**, where the approach is more centered on a pure prediction objective, most of the time by a "black box" without the need for an explicit interpretation. With the increase in the size of datasets (in the era of Big Data), algorithms have been developed in Python, in particular in the *scikit learn* library.

A common **objective of learning** is to built a prediction algorithm, minimizing a prediction error, with or without the constraint of interpretability of the algorithm. Contexts are diverse, whether the aim is to publish a research article in an academic journal or participating in a Kaggle-type competition or developing an industrial solution for example for recommendation systems, fraud detection, predictive maintenance algorithms ... The publication of a new learning method or new options of existing methods requires showing that it outperforms its competitors on a battery of examples, generally from the site hosted at the University of California Irvine *UCI Repository* [26]. The biases inherent in this approach are discussed in numerous articles (*e.g.* Hand; 2006) [20] and conferences (*e.g.* Donoho (2015) [13]. It is notable that the academic pressure of publication has caused an explosion in the number of methods and their variants. The analysis of *Kaggle* type competitions and their winning solutions is also very instructive. The pressure leads to combinations, even architecture of models, of such complexity (see e.g. Figure 1.1) that these solutions are concretely unusable for slight performance differences (3rd or 4th decimal).

Especially if the data are voluminous, the operational and "industrialized" solutions, necessarily robust and fast, are often satisfied with rather rudimentary methodological tools (see Donoho (2015) [13]).

This course proposes to address the wide variety of criteria and methods, their conditions of implementation, the choices to be made, in particular to op-

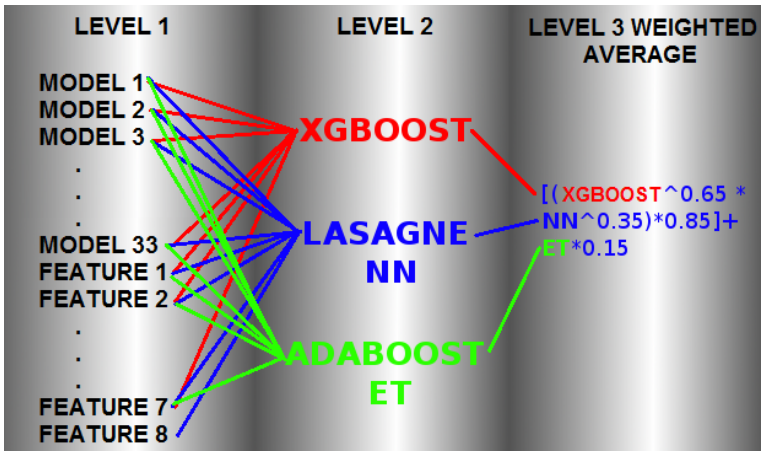


Figure 1.1: *Winning solution of a kaggle contest: Identify people who have a high degree of Psychopathy based on Twitter usage. Weighted combination of combinations (boosting, neural networks) of thirty three models (random forest, boosting, k nearest neighbors ...) and 8 new variables (*features*)*

timize the complexity of the models. It is also the opportunity to remind that robust and linear methods as well as old strategies (descending, ascending, step-by-step) or more recent (lasso) for the selection of linear or polynomial models should not be too quickly evacuated from academic or industrial practices.

2 Tutorials and datasets

Tutorials and practical works are important to illustrate the behavior and the performances of the studied methods or algorithms and to become more familiar with them. In addition to pedagogical examples simply illustrating the different methods, other full-scale examples allow to really assess the efficiency of the machine learning algorithms but also all the complexity of their implementation.

The analysis of these different usecases is presented in tutorials contained in **jupyter notebooks** in R or Python. They are available in the repository `github.com/wikistat`

We present here some of the datasets that will be considered.

Ozone dataset

This example, studied by Besse et al. (2007) [5] is a real situation whose objective is to predict, for the next day, the risk of exceeding the legal ozone concentration threshold in urban areas. The problem can be considered as a regression problem: the variable to explain is an ozone concentration, but also as a binary classification problem: exceeding or not the legal threshold. There are only 8 explanatory variables, one of them is already a prediction of ozone concentration but obtained by a deterministic fluid mechanics model (Navier and Stockes equations). This is an example of *statistical adaptation*. The deterministic forecast on the basis of a global grid (30 km) is improved locally, at

the scale of a city, by a statistical model including this deterministic prediction but also other variables such as concentration of nitrogen oxide and dioxide, temperature, wind speed and wind direction. The more complete description is given in the following tabular:

Ozone data set: 1041 observations of the following components:

JOUR	type of the day: public holiday (1) or not (0)
O3obs	Ozone concentration observed the next day at 17h. generally the maximum of the day
MOCAGE	Prediction of this pollution obtained by a deterministic model of fluid mechanics
TEMPE	Temperature forecast by Météo France for the next day 17h
RMH2O	Moisture ratio
NO2	Nitrogen dioxide concentration
NO	Concentration of nitric oxide
STATION	Location of the observation: Aix-en-Provence, Rambouillet, Munchhausen, Cadarache and Plan de Cuques
VentMOD	Wind force
VentANG	Orientation of the wind

This example, of both regression and binary classification, has pedagogical virtues which allow it to be used as a *red thread* to compare most methods.

Human Activity Recognition (HAR) dataset

The HAR dataset are Public data, which were acquired and described by Anguita et al. (2013) [3]. They are available on the *UCI repository* and they represent usecases of Human Activity Recognition from signal recordings (gyroscope, accelerometer) obtained with a smartphone. The dataset contains 9 signals per individual: the accelerations in x, y, and z, those by subtracting the natural gravity and the angular accelerations in x, y, and z obtained from the gyroscope. Each signal contains $p = 128$ measures sampled at 64 htz during 2s. 7352 samples for learning and 2947 for testing. The objectives: Activity recognition (6 classes) standing, sitting, lying, walking, walking upstairs or walking downstairs: this is a supervised classification problem.

The first step is to build machine learning algorithms from the "features"

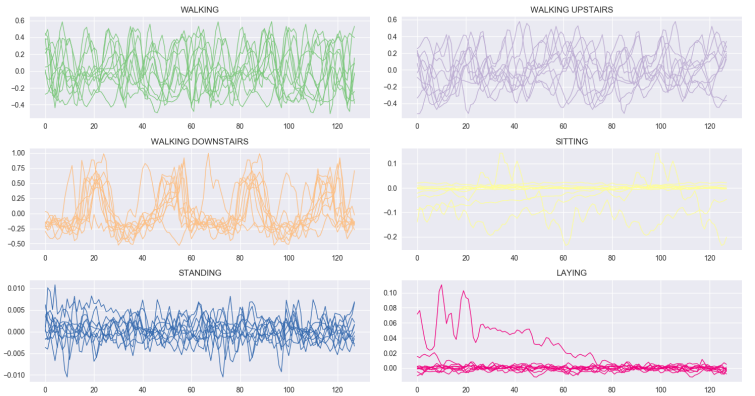


Figure 1.2: *Human activity recognition acceleration in y by class*

variables obtained by transformation of the raw data with signal processing techniques: $p = 561$ new variables (*features*) obtained in the time domain: min, max, means, variances, correlations... and in the frequency domain: largest, mean, energy per frequency band... The next step is to try to obtain the same performances directly on the raw data by algorithms for functional data such as 1D or 2D convolutional neural networks (High Dimensional and Deep Learning course).

MNIST dataset

This famous data set is available on Yann le Cun website. It is composed of a learning set with 60.000 handwritten digits, $28 \times 28 = 784$ pixels and a test set with 10.000 images. The images are labelled, this is therefore a supervised classification problem with 10 classes: $0, 1, \ldots, 9$. Buy transforming these images into vectors, we can apply classical methods such as k-nn, Random

Forests, neural networks... More appropriate algorithms, acting directly on images such as convolutional neural networks will be studied next year.

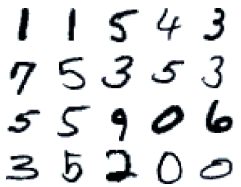


Figure 1.3: *MNIST some examples of handwritten digits*

3 Introduction to supervised learning

In the framework of **Supervised learning**, we have a **Learning sample** composed with observation data of the type **input/output**:

$$d_1^n = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$$

where, for $i = 1 \ldots n$, $\boldsymbol{x}_i = (x_i^1, \ldots, x_i^p) \in \mathcal{X}$ is a set of p explanatory variables and $y_i \in \mathcal{Y}$ is a response variable.
In this course, we consider supervised learning for real regression ($\mathcal{Y} \subset \mathbb{R}$) or for classification ($\mathcal{Y}$ finite). The explanatory variables $x^1, \ldots x^p$ can be qualitatives or quantitatives.

Objectives: From the learning sample, we want to

- **Estimate** the link between the input vector $\boldsymbol{x}$ (explanary variables) and the output y (variable to explain):

$$y = f(\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p).$$

- **Predict** the output y associated to a new entry $\boldsymbol{x}$.
- **Select** the important explanatory variables among $x^1, \ldots, x^p$.

We consider **supervised regression or classification problems**. We have a training data set with n observation points (or objects) $\boldsymbol{X}_i$ and their associated output Y_i (real value in regression, class or label in classification).
$\boldsymbol{d}^n$ corresponds to the observation of a random n-sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ with joint unknown distribution P on $\mathcal{X} \times \mathcal{Y}$.

A **prediction rule** is a measurable function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that associates the output $\hat{f}(\boldsymbol{x})$ to the input $\boldsymbol{x} \in \mathcal{X}$.
In order to quantify the quality of the prevision, we introduce a loss function.

DEFINITION 1. — *A measurable function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a* loss function *if $\ell(y, y) = 0$ and $\ell(y, y') > 0$ for $y \neq y'$.*

In real regression, it is natural to consider $\mathbb{L}^p$ ($p \geq 1$) losses

$$l(y, y') = |y - y'|^p.$$

If $p = 2$, the $\mathbb{L}^2$ loss is called "quadratic loss".
In classification, one can consider the consider the 0-1 loss defined, for all $y, y' \in \mathcal{Y}$ by

$$l(y, y') = \mathbf{1}_{y \neq y'}.$$

Since the 0-1 loss is not smooth, it may be useful to consider other losses that we will see in the classification courses.

The goal is to minimize the expectation of this loss function, leading to the notion of *risk*:

DEFINITION 2. — *Let f be a prediction rule defined from the learning sample $\boldsymbol{D}^n$. Given a loss function ℓ, the* **risk** *- or* **generalization error** *- of the*

prediction rule f is defined by

$$R_P(f) = \mathbb{E}_{(\boldsymbol{X},Y)\sim P}[l(Y, f(\boldsymbol{X}))],$$

where, in the above expression, $(\boldsymbol{X}, Y)$ is independent from the learning sample $\boldsymbol{D}^n$.

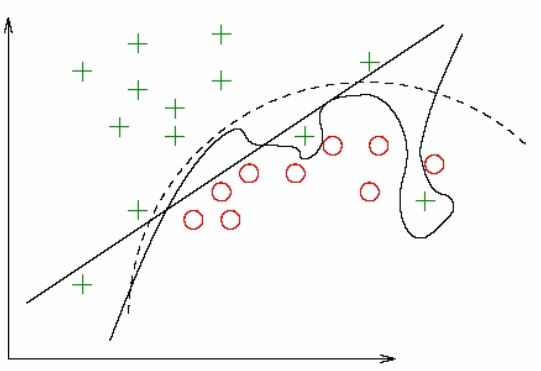
An accurate evaluation of the generalization error has two objectives:

- **Model selection:** selecting, among a collection of models (or prediction rules), the one with the smallest risk, realizing the best bias/variance trade-off.
- **Model assessment:** once the final model has been chosen, evaluating its generalization error on a new data set.

In practice, we have a training sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ with unknown joint distribution P, from which we construct a regression or classification rule. The aim is to find a "good" classification rule, in the sense that its risk is as small as possible. In order to evaluate a prediction rule, we have to estimate its risk.
A first natural idea to estimate the risk $R_P(f) = \mathbb{E}_{(\boldsymbol{X},Y)\sim P}[l(Y, f(\boldsymbol{X}))]$ is to consider its empirical estimator, called **empirical risk**, or **training error**:

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} l(Y_i, f(\boldsymbol{X_i})).$$

This is not a good idea: this estimator is optimistic and will under estimate the risk (or generalisation error) as illustrated in the following binary classification example, where three classification rules are compared.



Supervised binary classification: Complexity of the models.

The **generalization** performance of a learning procedure is related to its prediction capacity on a **new data set**, independent of the learning sample that was used to build the learning algorithm.

If we have enough data, the recommended approach is to divide randomly the dataset in two parts: the train sample and the test sample, the train sample being itself divided into a learning sample and a validation sample.

- **The learning sample** is used to train the models (generally by minimizing the training error).
- **The validation sample** is used for model selection: we estimate the generalization error of each model with the validation sample and we select the model with the smallest generalization error.
- **The test sample** is used for model assessment, to evaluate the risk of the final selected model.

It is generally recommended to take 50% of the data for the learning sample, 25% of the data for the validation sample and 25% of the data for the test sample.

Splitting the data set is not always a good solution, especially if its size is quite small. We will see in Chapter 2 several ways to estimate the generalization error.

4 Strategy for statistical learning

4.1 The steps of a statistical analysis

In a real situation, the initial preparation of the data (*data munging*: extraction, cleaning, verification, possible allocation of missing data, transformation ...) is the most thankless phase, the one that requires the most time, human resources and various skills: informatics, statistics and knowledge of the field of the data. This stage does not require major theoretical developments but rather a lot of common sense, experience and a good knowledge of the data. Once successfully completed, the modeling or learning phase can begin.

Systematically and also very schematically, the analysis, also called the *Data science* follows the steps described below for most fields of application.

1. Data extraction with or without sampling applied to structured databases (SQL) or not (NoSQL)
2. Visualization, exploration of the data for the detection of atypical values, errors or anomalies; study of distributions and correlation structures and search for transformations of variables, construction of new variables and / or representation in adapted bases (Fourier, spline, wavelets ...).
3. Taking into account missing data, by simple deletion or by imputation.
4. Random partition of the sample into a **train set** and a **test set** according to its size and choice of a loss function that will be used to estimate the prediction error.
5. **The train set** is separated into a **learning sample** and a **validation sample**. For each method considered: generalized linear model (Gaussian, binomial or Poisson), parametric (linear or quadratic) or nonparametric (k nearest neighbors), discrimination, neural network (perceptron), binary decision tree, support vectors machine, aggregation (*bagging, boosting, random forest*...)
   - Estimate the model with the **learning set** for given values of a parameter of *complexity*: number of variables, neighbors, leaves, neurons, penalization or regularization ...
   - optimization of this parameter (or these parameters) by minimizing the empirical loss on the **validation set**, or by cross-validation on the **train set** or the training error plus a penalty term.
6. Comparison of the previous optimal models (one per method) by estimating the prediction error on the **test set**.
7. Possible iteration of the previous approach or *Monte Carlo* cross-validation: if the test sample at step 4 is too small, the prediction error obtained at step 6 can be very dependent on this test sample. The Monte Carlo cross-validation approach consists in successive random partitions of the sample (train and test) to study the distribution of the test error for each model or at least take the mean of the prediction errors obtained from several Monte-Carlo iterations to ensure the robustness of the final selected model.
8. Choice of the "best" method according to its prediction error, its robustness but also its interpretability if necessary.

9. Re-estimation of the selected model on all the data.
10. Industrialization: implementation of the model on the complete data base.

The end of this process can be modified by building a combination of the different methods rather than selecting the best one. This is often the case with winning "gas factory" solutions in *Kaggle* competitions. This has also been theorized in two approaches leading to a *collaboration* between models: COBRA from Biau et al. (2016) [6] and *SuperLearner* from van der Laan et al. (2007) [39].

4.2 The methods or algorithms

We will see during this course the most widespread learning methods.

- In chapter 2, we will see how to estimate the prediction error of an algorithm. This is a crucial step to choose the "best" prediction rule, among a collection, and to evaluate the performances of the selected procedure.
- In chapter 3, make some reminders on linear models and logistic regression. In chapter 4, we introduce model selection for linear models via penalized criterion: Mallows CP, BIC, Ridge, Lasso...
- Linear methods for classification will be the subject of Chapter 5, where we will study the linear (and quadratic) discriminant analysis and the Linear Support Vector Machine (SVM). This chapter will be followed in Chapter 6, by the introduction of classification and regression algorithms based on kernel methods: Support Vector Machine (SVM) and Support Vector Regression, particularly adapted to analyse various kinds of data.
- We will then study Classification And Regression Trees (CART algorithm) in Chapter 7, and the aggregating methods and the Random Forests in Chapter 8.
- Neural networks will be introduced in Chapter 10. We will focus on multilayer perceptron, backpropagation algorithms, optimization algorithms, and provide an introduction to deep learning to will be completed next year by the study of the Convolutional Neural Networks.
- Finally, we will approach ethical aspects of statistical decisions and legal and societal impacts of AI.

Chapter 2

Risk estimation and Model selection

1 Introduction

1.1 Objectives

The performance of a model or algorithm is evaluated by a *risk* or *generalization error*. The measurement of this performance is very important since, on the one hand, it allows to operate a *model selection* in a family of models associated with the learning method we used and, on the other hand, it guides the *choice of the best method* by comparing each of the optimized models at the previous step. Finally, it provides a measure of the quality or even of the *confidence* that we can give to the prediction with the selected model.

Once the notion of statistical model or *prediction rule* is specified, the *risk* is defined from an associated *loss* function. In practice, this risk needs to be estimated and different strategies are proposed.

The main issue is to construct an *unbiased* estimator of this risk. The empirical risk (based on the training sample), also called the training error is *biased by optimism*, it underestimates the risk. If we compute an empirical estimator of the risk on a test sample (independent of the training sample), measuring the generalization capacity of the algorithm, we generally obtain higher values. If these new data are representative of the whole distribution of the data, we obtain an unbiased estimator of the risk. Three strategies are described to obtain *unbiased estimates of risk*:

1. a *penalisation* of the empirical risk
2. a split of the sample: train set and test set. The train set is itself decomposed into a leaning set to estimate the models for a given algorithm and a validation set to estimate the generalization error of each model in order to choose the best one, the test set is used to estimate the risk of each optimized model.
3. by simulation: cross validation, *bootstrap*.

The choice depends on several factors including the desired objective, the size of the initial sample, the complexity of the models, the computational complexity of the algorithms.

2 Risk and model selection

2.1 Loss function and risk

We consider **supervised regression or classification problems**. We have a training data set with n observation points (or objects) $\boldsymbol{X}_i$ and their associated output Y_i (real value in regression, class or label in classification).
$\boldsymbol{d}^n$ corresponds to the observation of the random n-sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ with unknown joint distribution P on $\mathcal{X} \times \mathcal{Y}$.

A **prediction rule** is a measurable function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that associates the output $\hat{f}(\boldsymbol{x})$ to the input $\boldsymbol{x} \in \mathcal{X}$. It depends on $\boldsymbol{D}^n$ and is thus random.
In order to quantify the quality of the prevision, we introduce a loss function.

DEFINITION 3. — *A measurable function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ is a* loss function *if $\ell(y, y) = 0$ and $\ell(y, y') > 0$ for $y \neq y'$.*

In real regression it is natural to consider $\mathbb{L}^p$ ($p \geq 1$) losses

$$\ell(y, y') = |y - y'|^p.$$

If $p = 2$, the $\mathbb{L}^2$ loss is called "quadratic loss".
In classification, one can consider the 0-1 loss defined, for all $y, y' \in \mathcal{Y}$ by

$$\ell(y, y') = \mathbf{1}_{y \neq y'}.$$

Since the 0-1 loss is not smooth, it may be useful to consider other losses.

Assuming that $Y \in \{1, 2, \ldots, K\}$, rather than providing a class, many classification algorithms provide estimation of the probability that the output Y belongs to each class, given the input $\boldsymbol{X} = x$, that is

$$\hat{f}_k(x) = \hat{P}(Y = k / \boldsymbol{X} = x), \forall k = 1, \ldots, K.$$

Then, the prediction rule generally assigns to the input x the class that maximizes the estimated probability that is

$$\hat{f}(x) = \text{argmax}_{k \in \{1,2,\ldots,K\}} \hat{f}_k(x).$$

In this setting, a loss function often used is the so-called **cross-entropy** (or negative log-likelihood). Minimizing this loss function is equivalent to maximizing the log-likelihood. It is defined as:

$$\ell(Y, \hat{f}(\boldsymbol{X})) = -\sum_{k=1}^{K} \mathbf{1}_{Y=k} \log(\hat{f}_k(\boldsymbol{X})).$$

In all cases, the goal is to minimize the expectation of the loss function, leading to the notion of *risk*.

DEFINITION 4. — *Let f be a prediction rule build on the learning sample $\boldsymbol{D}^n$. Given a loss function ℓ, the* **risk** *- or* **generalisation error** *- of f is defined by*

$$R_P(f) = \mathbb{E}_{(\boldsymbol{X},Y)\sim P}[\ell(Y, f(\boldsymbol{X}))],$$

where, in the above expression $(\boldsymbol{X}, Y)$ is independent from the learning sample $\boldsymbol{D}^n$.

Let $\mathcal{F}$ be the set of possible prediction rules. f^* is called an optimal rule if

$$R_P(f^*) = \inf_{f \in \mathcal{F}} R_P(f).$$

A natural question then arises: is it possible to build optimal rules ?

Case of real regression with $\mathbb{L}_2$ loss:

$$\mathcal{Y} = \mathbb{R}, \quad \ell(y, y') = (y - y')^2.$$

DEFINITION 5. — *We call* **regression function** *the function* $\eta^* : \mathcal{X} \to \mathcal{Y}$ *defined by*

$$\eta^*(\boldsymbol{x}) = \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}].$$

THEOREM 1. — *The regression function* $\eta^* : \boldsymbol{x} \mapsto \mathbb{E}[Y|\boldsymbol{X} = \boldsymbol{x}]$ *satisfies:*

$$R_P(\eta^*) = \inf_{f \in \mathcal{F}} R_P(f).$$

Case of real regression with $\mathbb{L}_1$ loss :

$$\mathcal{Y} = \mathbb{R}, \quad \ell(y, y') = |y - y'|.$$

THEOREM 2. — *The regression rule defined by* $\mu^*(\boldsymbol{x}) = median[Y|\boldsymbol{X} = \boldsymbol{x}]$ *verifies:*

$$R_P(\mu^*) = \inf_{f \in \mathcal{F}} R_P(f).$$

Case of classification with $0-1$ loss :

$$\ell(y, y') = \mathbf{1}_{y \neq y'}.$$

DEFINITION 6. — *We call* Bayes rule *any function* f^* *of* $\mathcal{F}$ *such that for all* $\boldsymbol{x} \in \mathcal{X}$,

$$\mathbb{P}(Y = f^*(\boldsymbol{x})|\boldsymbol{X} = \boldsymbol{x}) = \max_{y \in \mathcal{Y}} \mathbb{P}(Y = y|\boldsymbol{X} = \boldsymbol{x}).$$

THEOREM 3. — *If* f^* *is a Bayes rule, then* $R_P(f^*) = \inf_{f \in \mathcal{F}} R_P(f)$.

The definition of the optimal rules described above depends on the knowledge of the distribution P of $(\boldsymbol{X}, Y)$. In practice, we have a training sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ with joint unknown distribution P, from which we construct a regression or classification rule. The aim is to find a "good" classification rule, in the sense that its risk is as small as possible.

2.2 Minimisation of the empirical risk

In order to evaluate a prediction rule, we have to estimate its risk.
A first natural idea to estimate the risk $R_P(f) = \mathbb{E}_{(\boldsymbol{X},Y)\sim P}[l(Y, f(\boldsymbol{X}))]$ is to consider its empirical estimator, called **empirical risk**, or **training error**:

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, f(\boldsymbol{X_i})).$$

Nevertheless, this is not a good idea: this estimator is optimistic and will under estimate the risk (or generalisation error) as illustrated in the polynomial regression example presented in Figure 2.1.

The empirical risk (also called training error) is not a good estimate of the generalization error: it decreases as the complexity of the model increases. Hence minimizing the training error leads to select the most complex model, this leads to **overfitting**. Figure 2.2 illustrates the optimism of the training error, that underestimates the generalization error, which is estimated here on a test sample.

A first way to have a good criterion for model selection is to minimize the empirical risk **plus a penalty term**, the penalty term will penalize too complex model to prevent overfitting.

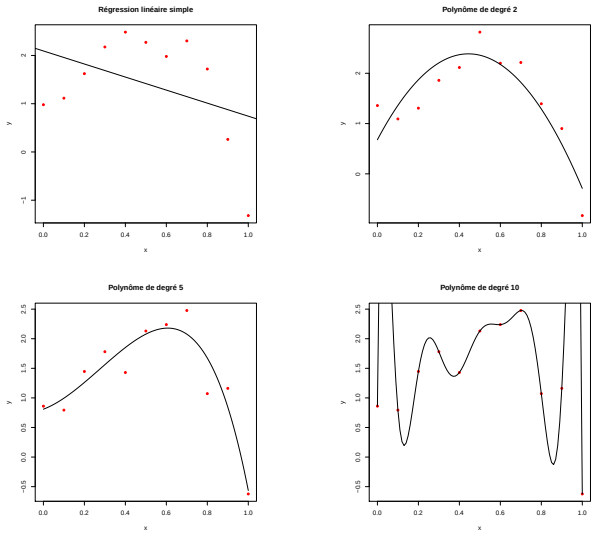


Figure 2.1: Polynomial regression: adjusted model are polynomials with respective degrees 1: $R^2 = 0.03$, 2: $R^2 = 0.73$, 5: $R^2 = 0.874$ and 10: $R^2 = 1$. The empirical risk is equal to 0 for the polynomial of degree $n-1$ (which has n coefficients) and passes through all the training points. Selecting a model which minimizes the empirical risk leads to overfitting.

2.3 Minimisation of the penalized empirical risk

Decomposition approximation/estimation (or bias/variance)

Let f^* such that $R_P(f^*) = \inf_f R_P(f)$, f^* is an optimal rule, also called an "*oracle*". From the training sample, the objective is to determine a model

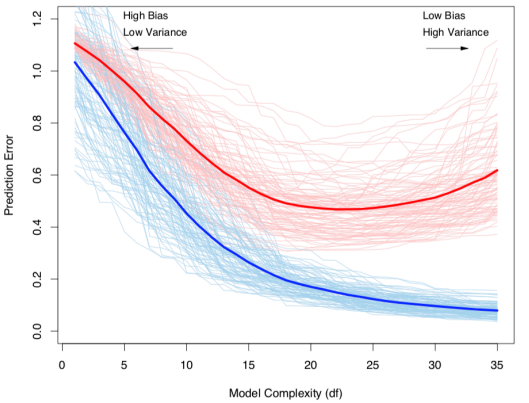


Figure 2.2: Behavior of training error (in blue) and test error (in red) as the complexity of the model increases. Source: "The elements of Statistical Learning", T. Hastie, R. Tibshirani, J. Friedman.

F for which the risk of the estimator $\hat{f}_F$, based on this model (for example by minimization of the empirical risk), is close to the one of the oracle

$$R_P(\hat{f}_F) - R_P(f^*) = \underbrace{\left\{ R_P(\hat{f}_F) - \inf_{f \in F} R_P(f) \right\}}_{\textbf{Estimation error}} + \underbrace{\left\{ \inf_{f \in F} R_P(f) - R_P(f^*) \right\}}_{\textbf{Approximation error}}$$

(Variance) ↗ (Bias) ↘ (dimension of F)

These two terms are of different natures. To evaluate them, we will use tools respectively from statistics and approximation theory.

The selection of a model $\hat{F}$ in a collection of models $\mathcal{C}$ for which the risk of the estimator $\hat{f}_{\hat{F}}(\boldsymbol{D}_n)$ is close to the one of the oracle will be obtained by the minimization of a penalized criterion of the type:

$$\hat{F} = \text{argmin}_{F \in \mathcal{C}} \{R_n(\hat{f}_F) + \text{pen}(F)\}.$$

In the above formula, a penalty is added to the empirical risk. The role of the penalty is to penalize models with "large" dimension, in order to avoid overfitting. The optimal choice of the penalty (according to the statistical models considered) is a very active research topic in statistics.

The more complex a model, the more flexible it is and can adjust to the observed data and therefore the smaller the bias. On the other hand, the variance increases with the number of parameters to be estimated and therefore with this complexity. The objective is to minimize the quadratic risk, which is a sum of the variance and the squared bias term. Hence, we are looking for the best compromise between the bias and the variance term: it is sometimes preferable to accept to bias the estimate as for example in *ridge* regression to reduce its variance.

Penalized criterion: Mallow's C_P

The Mallow's C_p (1973)[28] was historically the first penalized criterion, introduced for Gaussian linear model. It is based on the penalization of the least square criterion by a penalty which is proportional to the dimension of the model. It is based on the decomposition

$$R_P(\widehat{f}) = R_n(\widehat{f}) + \text{Optim}$$

which corresponds to the empirical risk plus a estimation of the bias corresponding to the optimism of the empirical risk. This optimism has to be estimated to obtain a better estimation of the risk. This criterion is expressed as follows

$$C_p = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + 2\frac{p}{n}\widehat{\sigma}^2$$

where p is the number of parameter of the model, n the number of observations and $\widehat{\sigma}^2$ is an estimation of the variance of the error.

In framework of a the linear model $Y = X\beta + \varepsilon$, for which this criterion was historically introduced, the expression becomes

$$C_p = \frac{1}{n}\sum_{i=1}^{n}(Y_i - (X\hat{\beta})_i)^2 + 2\frac{p}{n}\widehat{\sigma}^2,$$

where $\beta \in \mathbb{R}^p$ and σ^2 is an estimator of the variance of the variables ε_i's obtained by a model with large dimension (small bias). This last point is crucial for the quality of the criterion: it amounts to assume that the full model (with all the variables) is the "true" model, or at least a model with a small bias to allow a good estimation of σ^2.

The Figure 2.3 shows the behavior of the Mallow's C_p in the pedagogical example of polynomial regression. This criterions selects a polynomial with degree 3.

AIC, AIC$_c$, BIC

While Mallow's C_P is associated to the quadratic loss, Aikaike's Information Criterion (1974)[2] (AIC) is, more generally, related to the log-likelihood. It corresponds to the opposite of the empirical log-likelihood $\mathcal{L}$ plus a penalty term proportional to the dimension of the model:

$$\text{AIC} = -2\mathcal{L} + 2\frac{p}{n}.$$

The quantity $-2\mathcal{L}$ is also called *deviance*. One easily verifies that, in the Gaussian model with variance assumed to be known, the deviance and least square

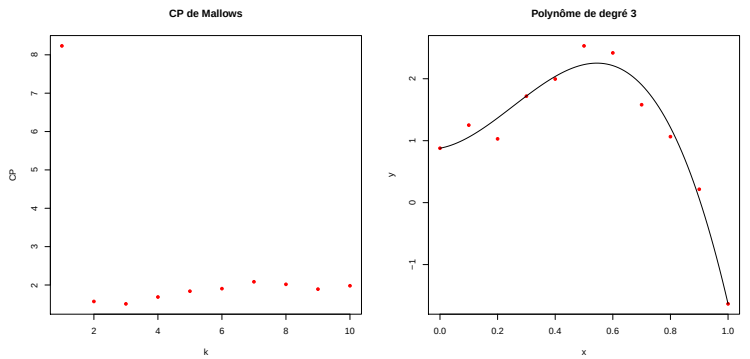


Figure 2.3: *Polynomial regression: Mallow's C_P against the degree of the polynomial: a polynomial with degree 3 is selected*

criterion coïncide. In this case, AIC is equivalent to C_P. A refined version of the AIC criterion, called corrected AIC is defined as

$$\text{AIC}_c = -2\mathcal{L} + \frac{n+p}{n-p-2}.$$

It is recommended for small samples sizes and asymptotically equivalent to AIC for large values of n.

Another criterion called BIC, (*Bayesian Information Criterion*) (Schwartz; 1978) [35] derives from Bayesian arguments. It is also based on the penalization of the negative log likelihood, but with a higher penalty than AIC:

$$\text{BIC} = -2\mathcal{L} + \log(n)\frac{p}{n}.$$

Since the factor 2 in the AIC criterion is here replaced by $\log(n)$, as soon as $n > e^2 \approx 7.4$, BIC penalizes more heavily complex models. The consequence is that BIC will generally select simpler model than AIC.

Whatever the chosen criterion, the strategy is to select a model minimizing this criterion, among a collection of possible models.

3 Estimation of the generalization error

Instead of minimizing a penalized criterion, other strategies for model selection consists in estimating the generalization error, either with data that where not used during the training phase, or by Bootstrap's methods.

The **generalization** performance of a learning procedure is related to its prediction capacity on a **new data set**, independent of the learning sample that was used to build the learning algorithm. Evaluating this performance is crucial to choose a learning method or model among several possible ones. It is also important to measure the quality of the ultimately chosen procedure. It is therefore crucial to estimate the generalization error of a learning algorithm $\hat{f}$: when the model becomes more and more complex, it is able to capture more complex underlying structures in the "true " model: the bias decreases, but at the same time, the estimation error increases, due to the increase of the variance. The "optimal" model is the one realizing **the best compromise between the bias term and the variance term** to give the smallest generalization error.

An accurate evaluation of the generalization error has two objectives:

- **Model selection:** selecting, among a collection of models (or prediction rules), the one with the smallest risk, realizing the best bias/variance trade-off.
- **Model assessment:** Once the final model has been chosen, evaluating its generalization error on a new data set.

We concentrate here on the first objective, assuming that we have a **test set** for

model assessment.

3.1 Estimation by cross-validation

As seen previously, it is crucial to evaluate the performances of an algorithm on data that were not use during the learning step. For this purpose, cross-validation methods are widely used. The main variations of this method are presented here.

Holdout cross-validation

If we have enough data in the training set, the recommended approach is to divide randomly the training set into a learning sample and a validation sample.

- **The learning sample** denoted $\boldsymbol{D}_1^{n_1}$ is used to train the models (generally by minimizing the training error).
- **The validation sample** denoted $\boldsymbol{D}_2^{n_2}$ is used to estimate the generalization error of each model by the quantity

$$\frac{1}{n_2} \sum_{(X_i,Y_i)\in \boldsymbol{D}_2^{n_2}} \ell(Y_i, f(\boldsymbol{X_i})).$$

It is generally recommended to take 75% of the training data for the learning sample, 25% of the data for the validation sample.

Often, taking only 75% of the data set to train the models may lead to bad performances, especially if we do not have too much data. Moreover, if the size of the validation set is small, the estimation of the generalization error will have a high variance and be highly dependent on this validation set. To prevent this problem, K **fold cross-validation** is widely used.

K-fold cross-validation

K**-fold cross-validation** is a widely used method to estimate the generalization error without splitting the training set as done in the previous section. Its different steps are the following:

- We split randomly the training data into K subsamples, with (almost) the same size ($K = 10$ generally).
- Each of the K folds will be successively used as a validation sample.
- When the fold k is the validation sample, we train a model with the $K - 1$ other folds, and we evaluate the loss function of this model on each element the fold k.
- This is done for $k = 1, \ldots, K$, and we compute a global estimation of the generalization error.

More precisely, assume that we have a n-sample $(\boldsymbol{X}_i, Y_i)_{1\leq i\leq n}$ and a collection of models $(\hat{f}_m, m \in \mathcal{M})$. We split the data into K folds.

- Let for all $i \in \{1, \ldots, n\}$, $\tau(i) \in \{1, \ldots, K\}$ denote the number of the fold containing the observation $(\boldsymbol{X}_i, Y_i)$.
- Let $\hat{f}_m^{(-k)}$ denote the model m trained with all the data, except the fold k.
- The cross-validation estimate of the generalization error of the model m is

$$CV(m) = \frac{1}{n} \sum_{i=1}^{n} \ell(Y_i, \hat{f}_m^{(-\tau(i))}(\boldsymbol{X}_i)).$$

- $CV(m)$ estimates the generalization error of the model m and we select the model which minimizes $CV(m)$.

Note that, if K is small (for example $K = 2$), each estimator $\hat{f}_m^{(-k)}$ is trained with around $n/2$ observations. Hence, these estimators are less accurate than an estimator built with n observations, leading to a greater variance in the estimation of the generalization error by cross-validation. When the number of folds $K = n$, the method is called leave-one-out cross-validation. This method has a low bias to estimate the generalization error, but a high variance since all the estimators $\hat{f}_m^{(-i)}$ are highly correlated. The computation time is also high for the leave-one-out method. This is why, in practice an intermediate choice such as $K = 10$ is often recommended. This is generally the default value in softwares.

Monte Carlo Cross-Validation

This method consists in iterating several times the random subdivision of the initial sample into a learning set and a validation set. The most simple way to apply Monte Carlo Cross-Validation is to iterate the holdout procedure. The advantage of this method is to provide an estimation of the whole distribution of the risk, for all considered methods. The disadvantage is the computational time.

The proportion between samples: learning and test, depends on the initial sample size in order to preserve a significant part to the learning sample. The number B of iterations depends on the computation ressources. The smaller the initial sample size is, the less "independent" are the error evaluations and therefore the reduction in variance obtained at the end by the mean.

This strategy can also be coupled with K-fold cross-validation as described in the Algorithm 2. An example is presented in Figure 2.4.

Algorithm 1 Monte Carlo Cross-Validation

for k=1 à B **do**
Split randomly the sample into two parts: *training* set and *test* set with a prescribed proportion

for models *in* list of models **do**
Estimate the parameters of the current model with the training set.
Compute the test error by the empirical risk on the test set.
end for
end for
For each model, compute the mean of the B test errors and draw the boxplots of the distributions of these errors.

Algorithm 2 Monte Carlo K-fold Cross-Validation

for k=1 to B **do**
Split randomly the sample into two parts: *training* set and *test* set with a prescribed proportion.

for method *in* list of methods **do**
Optimise the complexity (or tuning parameters) of the method by *K-fold cross-validation*.
Estimate the parameters of the optimized model for this method with the training set.
Compute the test error by the empirical risk on the test set for the optimized model of the current method.
end for
end for
For each method, compute the mean of the B test errors and draw the boxplots of the distributions of these errors.

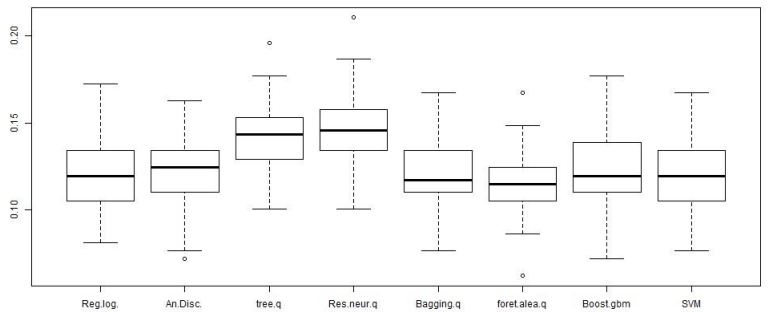


Figure 2.4: Boxplot of the test errors for various methods optimized by Monte Carlo K-fold Cross-Validation on Ozone data set

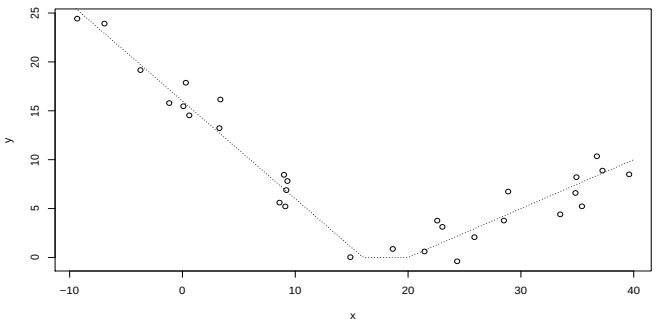


Figure 2.5: Original data

3.2 Estimation by Bootstrap

Let us first describe the Bootstrap, before showing how it can be used to estimate the extra-sample prediction error. Suppose we have a training data set $\mathbf{Z} = \{z_1, \dots z_n\}$, with $z_i = (x_i, y_i)$ and a model to be fitted on these data. We denote by $\hat{f}$ the model fitted with the sample $\mathbf{Z}$. The principle of the bootstrap is to randomly draw datasets of size n with replacement from the original sample $\mathbf{Z}$. Conditionally on $\mathbf{Z}$, all these draws are independent. Figures 2.6 and 2.7 show two bootstrap samples from the original dataset presented in Figure 2.5.

We draw B bootstrap samples (for example $B = 500$) that we denote $(\mathbf{Z}^{*b}, b = 1, \dots, B)$. We fit the model with each of these bootstrap samples. We denote $\hat{f}^{*b}$ the model fitted with the sample $\mathbf{Z}^{*b}$. How can we use all these predictors to estimate the prediction error of $\hat{f}$? A first idea would be to consider the following estimator:

$$\widehat{\text{Err}}_{boot} = \frac{1}{B}\frac{1}{n}\sum_{b=1}^{B}\sum_{i=1}^{n} \ell(y_i, \hat{f}^{*b}(x_i)),$$

measuring the mean, over the B bootstrap predictors, of the error on the training sample $\mathbf{Z}$. However, we easily see that this is not a good estimate of the generalization error since the bootstrap samples and the original sample have many observations in common. Hence, this estimator will be too optimistic: it will underestimate the generalization error. A better idea is to exploit the fact that each bootstrap sample does not contain all the observations of the original sample. Namely, we have

$$\mathbb{P}\left(\text{Observation } z_i \notin \text{ bootstrap sample } b\right) = \left(1 - \frac{1}{n}\right)^n \approx \frac{1}{e} = 0.368.$$

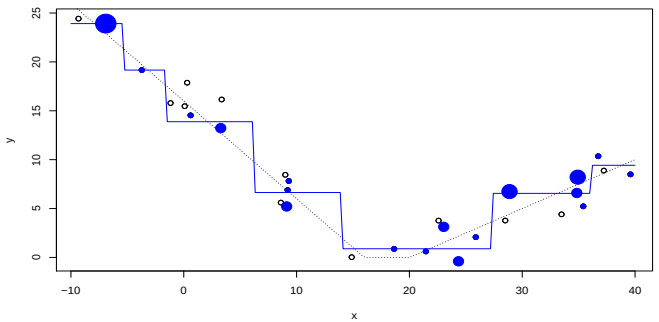


Figure 2.6: Bootstrap sample n^o1 (in blue), and corresp. prediction with tree. The point size is proportional to the number of replicates.

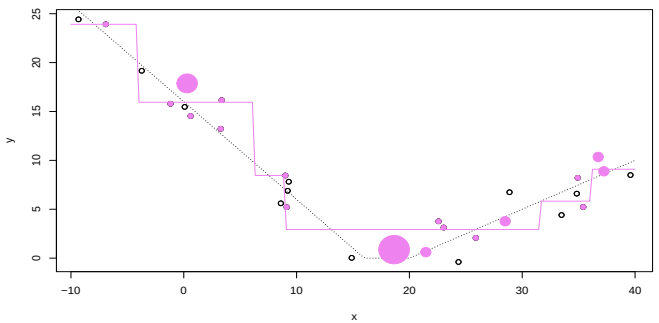


Figure 2.7: Bootstrap sample n^o2 (in violet), and corresp. prediction with tree. The point size is proportional to the number of replicates.

Mimicking the idea of cross-validation, we denote by C^{-i} the set of indices b in $\{1, \dots B\}$ such that $\mathbf{Z}^{*b}$ does not contain the observation z_i, and we introduce the estimator

$$\widehat{\text{Err}}_{oob} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}}\ell(y_i, \hat{f}^{*b}(x_i)).$$

This estimator is called the out-of-bag estimator. If B is large enough, then for all i , $|C^{-i}| \neq 0$. Otherwise, the observation i for which $|C^{-i}| = 0$ can be removed from the above formula. This estimator uses extra sample observation to estimate the error of each predictor $\hat{f}^{*b}$, avoiding the overfitting problem encountered by $\widehat{\text{Err}}_{boot}$. Nevertheless, in expectation, each bootstrap sample contains $0.632n$ observations, which is less that $2n/3$ and we would like to estimate the generalization error of a predictor $\hat{f}$ built with the n observations of the original sample $\mathbf{Z}$. Each bootstrap predictors $\hat{f}^{*b}$ will be less accurate than $\hat{f}$ since it is built with a smaller sample size. This induces a bias in the estimation of the generalization error of $\hat{f}$ by $\widehat{\text{Err}}_{oob}$. To correct this bias, the ".632 bootstrap estimator " has been introduced by Efron and Tibshirani (1997) [14]. It is defined by

$$\widehat{\text{Err}}^{(.632)} = .368\bar{err} + .632\widehat{\text{Err}}_{oob},$$

where $\bar{err}$ is the training error of $\hat{f}$. This estimator is problematic in overfitting situation, and a correction has been proposed in this case. It is called the *.632+bootstrap* (see Hastie et al. [21] p. 220 for more details).

Remarks.

1. All the estimators proposed to estimate the generalization error are asymptotically equivalent, and it is not possible to know which method will be more precise for a fixed sample size n.

2. The boostrap is time consuming and more complicated. It is less used in practice. Nevertheless, it plays a central role in recent methods of aggregation, involving the bagging (for bootstrap aggregating) such as random forests as we will see in Chapter 8 .

3. In conclusion, the estimation of a generalization error is delicate, and it is recommended to consider the same estimator to compare two prediction methods and to be very careful, without theoretical justification, to use one of these estimation to certify an algorithm. For this last purpose, the use of a test sample, with sufficiently large size, would be recommended.

We will end this chapter by presenting the ROC curves, that are used to compare the relative performances of several binary classification methods.

4 Discrimination and ROC curves

For a two class classification problem: $\mathcal{Y} = \{0, 1\}$, prediction methods often provide an estimator of $\mathbb{P}(Y = 1|\boldsymbol{X} = \boldsymbol{x})$. Then, a natural prediction is to affect the observation $\boldsymbol{x}$ to the class 1 if

$$\hat{\mathbb{P}}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) > \frac{1}{2}.$$

This gives a symmetric role to classes 0 and 1, which is sometimes not desirable (health context, for instance). The idea is to parameterize the decision by a new **threshold parameter** s:

$$\hat{\mathbb{P}}(Y = 1|\boldsymbol{X} = \boldsymbol{x}) > s \quad \Leftrightarrow \quad \boldsymbol{x} \text{ belongs to class } 1$$

s should be chosen according to policy decision, typically a tradeoff between the rate of true positive and false positive.

Confusion matrix

Given a threshold s, we use the prediction rule: if $\hat{\mathbb{P}}(Y_i = 1|\boldsymbol{X} = \boldsymbol{x_i}) > s$, then $\widehat{Y}_i = 1$, else $\widehat{Y}_i = 0$.

The *confusion* matrix crosses the modalities of the predicted variable for a threshold value s with those of the observed variable in a contingency table:

Prediction	Observation		Total
	$Y_i = 1$	$Y_i = 0$	
$\widehat{y}_i = 1$	$n_{11}(s)$	$n_{10}(s)$	$n_{1+}(s)$
$\widehat{y}_i = 0$	$n_{01}(s)$	$n_{00}(s)$	$n_{0+}(s)$
Total	n_{+1}	n_{+0}	n

In classic situations of medical diagnosis, marketing, pattern recognition, signal detection ... the following main quantities are considered:

- Number of positive conditions $P = n_{+1}$
- Number of negative conditions $N = n_{+0}$
- True positives $TP = n_{11}(s)$ ($\widehat{Y}_i = 1$ et $Y_i = 1$)
- True negatives $TN = n_{00}(s)$ ($\widehat{Y}_i = 0$ et $Y_i = 0$)
- False negatives $FN = n_{01}(s)$ ($\widehat{Y}_i = 0$ et $Y_i = 1$)
- False positives $FP = n_{10}(s)$ ($\widehat{Y}_i = 1$ et $Y_i = 0$)
- *Accuracy* and error rate: $ACC = \frac{TN+TP}{N+P} = 1 - \frac{FN+FP}{N+P}$
- True positive rate or *sensitivity, recall* $TPR = \frac{TP}{P} = 1 - FNR$
- True negative rate or *specificity, selectivity* $TNR = \frac{TN}{N} = 1 - FPR$

- Precision or *positive predictive value* $PPV = \frac{TP}{TP+FP} = 1 - FDR$
- False positive rate $FPR = \frac{FP}{N} = 1 - TNR$
- False negative rate $FNR = \frac{FN}{P} = 1 - TPR$
- False discovery rate $FDR = \frac{FP}{FN+TN}$,
- F_1 score or harmonic mean of precision and sensitivity

$$F_1 = 2 \times \frac{PPV \times TPR}{PPV + TPR} = \frac{2 \times TP}{2 \times TP + FP + FN}.$$

- $F_\beta (\beta \in \mathbb{R}^+)$ score,

$$F_\beta = (1 + \beta^2) \frac{PPV \times TPR}{\beta^2 PPV + TPR}.$$

The notions of *specificity* and *sensitivity* come from signal theory; their values depend directly on the threshold s. By increasing s, the sensitivity decreases while the specificity increases. A good model combines high sensitivity and high specificity for signal detection.

The last criterion F_β makes it possible to weight between specificity and sensitivity by taking into account the importance or the cost of false positives. The smaller β, the more expensive false positives are compared to false negatives.

By analogy with the first and second kind errors for testing procedures, we consider the two following quantities that will be used to draw the ROC curve.

- The False Positive Rate:

$$FPR(s) = \frac{\sharp\{i, \hat{Y}_i = 1, Y_i = 0\}}{\sharp\{i, Y_i = 0\}}.$$

- The True Positive Rate:

$$TPR(s) = \frac{\sharp\{i, \hat{Y}_i = 1, Y_i = 1\}}{\sharp\{i, Y_i = 1\}}.$$

The ROC curve plots TPR(s) versus FPR(s) for all values of $s \in [0, 1]$.
We illustrate the construction of a ROC curve for a naïf example of logistic regression in dimension 1 in Figure 2.8.

By making the threshold s vary in $[0, 1]$, we obtain the complete ROC curve presented in Figure 2.9

How to use ROC curve to select classifiers ? The "ideal" Roc curve corresponds to FPR=0 and TPR =1 (no error of classification).
We would like to use ROC curve to compare several classification rules, but generally, the curves will intersect as shown in Figure 2.9 The **AUC: Area Under the Curve** is a criterion which is often used to compare several classification rules.
In order to compare several methods with various complexity, the ROC curves should be estimated on a test sample, they are indeed optimistic on the learning sample.

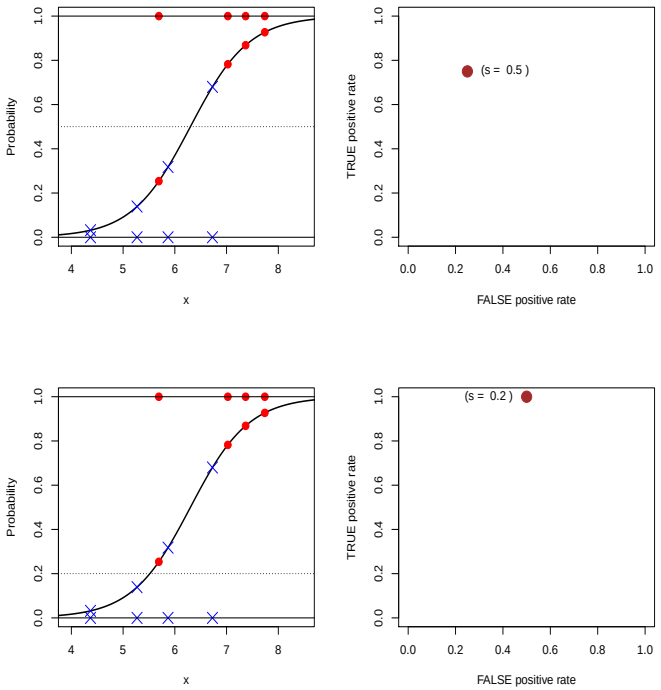


Figure 2.8: Points in the ROC curve obtained for $s = 0.5$ and $s = 0.2$

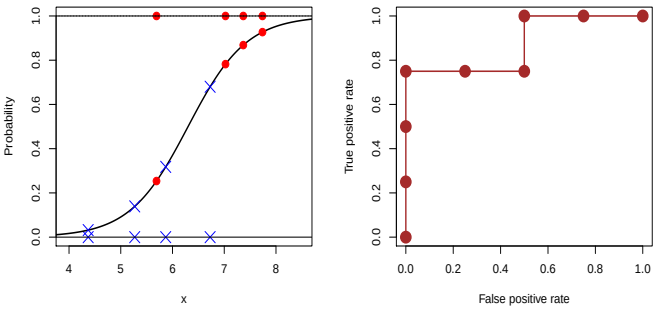


Figure 2.9: ROC curve

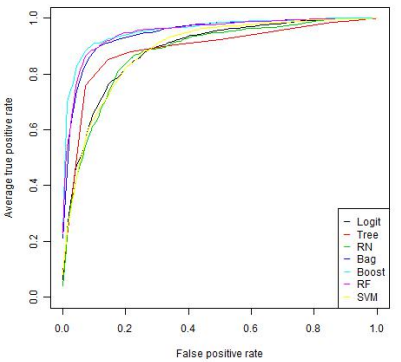


Figure 2.10: ROC curves for several classification rules on bank data

Chapter 3

Linear models

1 Introduction

The linear regression model is the simplest model to study multidimensional data. It assumes that the regression function $\mathbb{E}(\mathbf{Y}/\mathbf{X})$ is linear in the input (or explanatory) variables $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p$. Although very simple, these models are still widely used, because they are very interpretable and often provide an adequate description on the influence of the input variables to the output. For small sample sizes n (with respect to the number of variables p), or when the signal to noise ratio is high, they often outperform more complex models. Furthermore, it is possible to use linear models with nonlinear transformations of the variables, which considerably enlarges the scope of these models. In high dimensional framework, when p is possibly larger than n, model selection for linear models has been this past twenty years and is still a very active field of research in statistics. This will be the topic of Chapter 4. The aim of this chapter is to make some reminders on linear model for regression and logistic regression for classification.

2 The Linear model

2.1 The model

We have a quantitative variable $\mathbf{Y}$ *to explain* (or response variable) which is related with p variables $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p$ called *explanatory variables* (or regressors, or input variables).

The data are obtained from the observation of a n sample of $\mathbb{R}^{(p+1)}$ vectors :

$$(x_i^1, \ldots, x_i^j, \ldots, x_i^p, y_i) \quad i = 1, \ldots, n.$$

We assume in a first time that $n > p + 1$. In *the linear model*, the regression function $\mathbb{E}(\mathbf{Y}/\mathbf{X})$ is linear in the input (or explanatory) variables $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p$. We assume for the sake of simplicity that the regressors are deterministic. In this case, this means that $\mathbb{E}(\mathbf{Y})$ is linear in the explanatory variables $\{\mathbf{1}, \boldsymbol{X}^1, \ldots, \boldsymbol{X}^p\}$ where $\mathbf{1}$ denotes the $\mathbb{R}^n$-vector with all components equal to 1. The linear model is defined by:

$$Y_i = \beta_0 + \beta_1 X_i^1 + \beta_2 X_i^2 + \cdots + \beta_p X_i^p + \varepsilon_i \quad i = 1, 2, \ldots, n$$

with the following assumptions :

1. The random variables ε_i are independent and identically distributed (i.i.d.) ; $\mathbb{E}(\varepsilon_i) = 0, Var(\varepsilon_i) = \sigma^2$.

2. The regressors $\boldsymbol{X}^j$ are assumed to be deterministic **or** the errors $\boldsymbol{\varepsilon}$ are independent of $(\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p)$. In this case, we have :

   $E(\mathbf{Y}|\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p) = \beta_0 + \beta_1 \boldsymbol{X}^1 + \beta_2 \boldsymbol{X}^2 + \cdots + \beta_p \boldsymbol{X}^p$ and $Var(\mathbf{Y}|\boldsymbol{X}^1, \ldots, \boldsymbol{X}^p) = \sigma^2$.

3. The unknown parameters $\beta_0, \ldots, \beta_p$ are supposed to be constant.

4. It is sometimes assumed that the errors are Gaussian: $\boldsymbol{\varepsilon} = [\varepsilon_1 \cdots \varepsilon_n]' \sim \mathcal{N}_n(0, \sigma^2 \mathbf{I_n})$. The variables ε_i are then i.i.d. $\mathcal{N}(0, \sigma^2)$.

The explanatory variables are given in the matrix $\mathbf{X}(n \times (p+1))$ with general term X_i^j, the first column contains the vector $\mathbf{1}$ ($X_0^i = 1$). The regressors $\boldsymbol{X}^j$ can be quantitative variables, nonlinear transformation of quantitative variables (such as log, exp, square ..), interaction between variables $\boldsymbol{X}^j = \boldsymbol{X}^k.\boldsymbol{X}^l$, they can also correspond to qualitative variables: in this case the variables $\boldsymbol{X}^j$ are indicator variables coding the different levels of a factor (we remind that we need identifiability conditions in this case).
The response variable is given in the vector $\mathbf{Y}$ with general term Y_i. We set $\boldsymbol{\beta} = [\beta_0 \beta_1 \cdots \beta_p]'$, which leads to the matricial formulation of the linear model:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

As a practical example, we consider the **Ozone data set.**
The data frame has 1041 observations of the following components:

JOUR	type of the day ; public holiday(1) or not (0)
O3obs	Ozone concentration observed the next day at 17h., generally the maximum of the day
MOCAGE	Prediction of this pollution obtained by a deterministic model of fluid mechanics
TEMPE	Temperature forecast by MétéoFrance for the next day 17h
RMH2O	Moisture ratio
NO2	Nitrogen dioxide concentration
NO	Concentration of nitric oxide
STATION	Location of the observation: Aix-en-Provence, Rambouillet, Munchhausen, Cadarache and Plan de Cuques
VentMOD	Wind force
VentANG	Orientation of the wind.

We denote by Y the variable (**O3obs**) to explain. We set $X^1, \ldots X^p$ for the explanatory variables (**MOCAGE** , **TEMPE**, **JOUR** ..). The variables are quantitative (**MOCAGE** , **TEMPE** , ...), or qualitative (**JOUR**, **STATION**).
We consider the linear model:

$$Y_i = \beta_0 + \beta_1 X_i^1 + \beta_2 X_i^2 + \ldots + \beta_p X_i^p + \varepsilon_i,\ 1 \leq i \leq n,$$

For the qualitative variables, we consider indicator functions of the different levels of the factor, and introduce some constraints for identifiability. By default, in R, the smallest value of the factor are set in the reference.
This is an analysis of covariance model (mixing quantitative and qualitative variables).

2.2 Estimation of the parameters

Least square estimators

The regressors $\boldsymbol{X}^j$ are observed, the unknown parameters of the model are the vector $\boldsymbol{\beta}$ and σ^2. $\boldsymbol{\beta}$ is estimated by minimizing the residuals sum of square or equivalently, assuming that the errors are Gaussian, by maximisation of the likelihood.

We minimise with respect to the parameter $\boldsymbol{\beta} \in \mathbb{R}^{p+1}$ the criterion :

$$\begin{aligned}\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i^1 - \cdots - \beta_p X_i^p)^2 &= \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 \\ &= (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})'(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) \\ &= \mathbf{Y}'\mathbf{Y} - 2\boldsymbol{\beta}'\mathbf{X}'\mathbf{Y} + \boldsymbol{\beta}'\mathbf{X}'\mathbf{X}\boldsymbol{\beta}.\end{aligned}$$

Derivating the last equation, we obtain the *" normal equations"* :

$$2(\mathbf{X}'\mathbf{Y} - \mathbf{X}'\mathbf{X}\boldsymbol{\beta}) = 0$$

The solution is indeed a minimiser of the criterion since the Hessian $2\mathbf{X}'\mathbf{X}$ is positive semi definite (the criterion is convex) .

We make the additional assumption that the matrix $\mathbf{X}'\mathbf{X}$ is invertible, which is equivalent to the fact that the matrix $\mathbf{X}$ has full rank $(p+1)$ and so that there is no collinearity between the columns of $\mathbf{X}$ (the variables). Under this assumption, the estimation of $\boldsymbol{\beta}$ is give by :

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

and the predicted values of $\mathbf{Y}$ are :

$$\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y}$$

where $\mathbf{H} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ is called the "*hat matrix*" ; which puts a "hat" on $\mathbf{Y}$. Geometrically, it corresponds to the matrix of orthogonal projection in $\mathbb{R}^n$ onto the subspace Vect$(\mathbf{X})$ generated by the columns of $\mathbf{X}$.

Remark. — We have assumed that $\mathbf{X}'\mathbf{X}$ is invertible, which means that the columns of $\mathbf{X}$ are linearly independent. If it is not the case, this means that the application $\boldsymbol{\beta} \mapsto \mathbf{X}\boldsymbol{\beta}$ is not injective, hence the model is not identifiable and $\boldsymbol{\beta}$ is not uniquely defined. Nevertheless, even in this case, the predicted values $\widehat{\mathbf{Y}}$ are still defined as the projection of $\mathbf{Y}$ onto the space generated by the columns of $\mathbf{X}$, even if there is not a unique $\widehat{\boldsymbol{\beta}}$ such that $\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\boldsymbol{\beta}}$. In practice, if $\mathbf{X}'\mathbf{X}$ is not invertible (which is necessarily the case in high dimension when the number of variables p is larger than the number of observations n - since p vectors of $\mathbb{R}^n$ are necessarily linearly dependent), we have to remove variables from the model or to consider other approches to reduce the dimension (*Ridge*, Lasso, PLS ...) that we will developed in the next chapters.

We define the vector of residuals as:

$$\mathbf{e} = \mathbf{Y} - \widehat{\mathbf{Y}} = \mathbf{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}} = (\mathbf{I} - \mathbf{H})\mathbf{Y}$$

This is the orthogonal projection of $\mathbf{Y}$ onto the subspace Vect$(\mathbf{X})^{\perp}$ in $\mathbb{R}^n$. The variance σ^2 is estimated by

$$\widehat{\sigma}^2 = \frac{\|\mathbf{e}\|^2}{n-p-1} = \frac{\left\|\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right\|^2}{n-p-1}.$$

Properties of the least square estimator

THEOREM 4. — *Assuming that*

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

with $\boldsymbol{\varepsilon} \sim \mathcal{N}_n(0, \sigma^2\mathbf{I_n})$*, we obtain that* $\widehat{\boldsymbol{\beta}}$ *is a Gaussian vector:*

$$\widehat{\boldsymbol{\beta}} \sim \mathcal{N}_{p+1}(\boldsymbol{\beta}, \sigma^2(X'X)^{-1}).$$

In particular, the components of $\widehat{\boldsymbol{\beta}}$ *are Gaussian variables:*

$$\widehat{\boldsymbol{\beta}}_j \sim \mathcal{N}(\boldsymbol{\beta}_j, \sigma^2(X'X)^{-1}_{j,j}).$$

$$\hat{\sigma}^2 \sim \frac{\sigma^2}{n-(p+1)}\chi^2_{(n-(p+1))}$$

and is independent of $\hat{\beta}$.

Exercise. — Prove Theorem 4

$\widehat{\boldsymbol{\beta}}$ is a linear estimator of $\boldsymbol{\beta}$ (it is a linear transformation of the observation $\mathbf{Y}$) and it is unbiased. One can wonder if it has some optimality property. This is indeed the case: the next theorem, called the *Gauss-Markov* theorem, is very famous in statistics. It asserts that the least square estimator $\widehat{\boldsymbol{\beta}}$ has the smallest variance among all linear unbiased estimator of $\boldsymbol{\beta}$.

THEOREM 5. — *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *two matrices. We say that* $\mathbf{A} \preceq \mathbf{B}$ *if* $\mathbf{B} - \mathbf{A}$ *is positive semi-definite. Let* $\widetilde{\boldsymbol{\beta}}$ *a linear unbiased estimator of* $\boldsymbol{\beta}$*, with variance-covariance matrix* $\widetilde{\mathbf{V}}$*. Then,* $\sigma^2(\mathbf{X}'\mathbf{X})^{-1} \preceq \widetilde{\mathbf{V}}$.

Exercise. — Prove the Gauss-Markov theorem.

Theorem 5 shows that the estimator $\widehat{\boldsymbol{\beta}}$ is the best among all linear unbiased estimator of $\boldsymbol{\beta}$, nevertheless, in the next section, we will see that it can be preferable to consider biased estimator, if they have a smaller variance than $\widehat{\boldsymbol{\beta}}$, to reduce the quadratic risk. This will be the case for the Ridge, Lasso, PCR, or PLS regression.

Confidence intervals

One can easily deduce from Theorem 4 that

$$\frac{\widehat{\boldsymbol{\beta}}_j - \boldsymbol{\beta}_j}{\sqrt{\hat{\sigma}^2 (X'X)^{-1}_{i,i}}} \sim \mathcal{T}_{(n-(p+1))}$$

follows a Student distribution with $n-(p+1)$ degrees of freedom. This allows to build confidence intervals and tests for the parameters $\boldsymbol{\beta}_j$. The following interval is a 0.95 confidence interval for $\boldsymbol{\beta}_j$:

$$[\widehat{\boldsymbol{\beta}}_j - t_{n-(p+1),0.975}\sqrt{\hat{\sigma}^2 (X'X)^{-1}_{j,j}}, \widehat{\boldsymbol{\beta}}_j + t_{n-(p+1),0.975}\sqrt{\hat{\sigma}^2 (X'X)^{-1}_{j,j}}].$$

In order to test that the variable associated to the parameter $\boldsymbol{\beta}_j$ has no influence in the model, hence H_0: $\boldsymbol{\beta}_j = 0$ contre H_1: $\boldsymbol{\beta}_j \neq 0$, we reject the null hypothesis at the level 5% if 0 does not belong to the previous confidence interval.

Exercise. — Recover the construction of the confidence intervals.

Test of significance of a variable

We recall the linear model

$$Y_i = \beta_0 + \beta_1 X_i^1 + \beta_2 X_i^2 + \cdots + \beta_p X_i^p + \varepsilon_i \quad i = 1, 2, \ldots, n$$

We want to test if the variable X^j is significant in the model or not, which is equivalent to test the nullity of the parameter β_j.
We test H_0: $\beta_j = 0$ against H_1: $\beta_j \neq 0$.
Under the hypothesis H_0,

$$T_j = \frac{\widehat{\boldsymbol{\beta}}_j}{\sqrt{\hat{\sigma}^2 (X'X)^{-1}_{j,j}}} \sim \mathcal{T}_{(n-(p+1))}.$$

The p-value of the test is defined as

$$\mathbb{P}_{H_0}(|T_j| > |T_j|_{obs}) = \mathbb{P}(|\mathcal{T}_{(n-(p+1))}| > |T_j|_{obs}),$$

where $|T_j|_{obs}$ is the observed value for the variable $|T_j|$ with our data. If the p-value is very small, then it is unlikely that $|T_j|_{obs}$ is obtained from a Student distribution with $n-(p+1)$ degrees of freedom, hence we will reject the hypothesis H_0, and conclude that the variable X^j is significant. We fix some level α (generally 5%) for the test . If p-value $< \alpha$, we reject the nullity of β_j and conclude that the variable X^j is significant in the model. One easily prove that the probability to reject H_0 when it is true (i.e. to conclude that the variable X^j is significant when it is not) is less than the level α of the test.

On the example of the Ozone data set, the software R gives the following output, with the default constraints of R:

Coefficients	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-33.43948	6.98313	-4.789	1.93e-06 ****
JOUR1	0.46159	1.88646	0.245	0.806747
MOCAGE	0.37509	0.03694	10.153	< 2e-16 ***
TEMPE	3.96507	0.22135	17.913	< 2e-16 ***
...	...	...	...	...

Residual standard error: 27.83 on 1028 degrees of freedom

2.3 Prediction

As mentioned above, the vector of predicted values is

$$\widehat{\mathbf{Y}} = \mathbf{X}\widehat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y} = \mathbf{H}\mathbf{Y}.$$

This corresponds to the predicted values at the observation points. Based on the n previous observations, we may be interested with the prediction of the response of the model for a new point: $\mathbf{X_0}' = (1, {X_0}^1, \ldots, {X_0}^p)$:

$$Y_0 = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 X_0^1 + \boldsymbol{\beta}_2 X_0^2 + \ldots + \boldsymbol{\beta}_p X_0^p + \varepsilon_0,$$

where $\varepsilon_0 \sim \mathcal{N}(0, \sigma^2)$. The predicted value is

$$\widehat{Y}_0 = \widehat{\boldsymbol{\beta}}_0 + \widehat{\boldsymbol{\beta}}_1 {X_0}^1 + \ldots \widehat{\boldsymbol{\beta}}_p {X_0}^p = \mathbf{X_0}'\widehat{\boldsymbol{\beta}}.$$

We derive from Theorem 4 that

$$\mathbb{E}(\widehat{Y}_0) = \mathbf{X_0}'\boldsymbol{\beta} = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 X_0^1 + \boldsymbol{\beta}_2 X_0^2 + \ldots + \boldsymbol{\beta}_p X_0^p$$

and that $\widehat{Y}_0 \sim \mathcal{N}(\mathbf{X_0}'\boldsymbol{\beta}, \sigma^2 \mathbf{X_0}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X_0})$. We can deduce a confidence interval for the mean response $\mathbf{X_0}'\boldsymbol{\beta}$ at the new observation point $\mathbf{X_0}$:

$$\left[\mathbf{X_0}'\widehat{\boldsymbol{\beta}} - t_{n-(p+1),0.975}\hat{\sigma}\sqrt{\mathbf{X_0}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X_0}},\right.$$

$$\left.\mathbf{X_0}'\widehat{\boldsymbol{\beta}} + t_{n-(p+1),0.975}\hat{\sigma}\sqrt{\mathbf{X_0}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X_0}}\right].$$

A prediction interval for the response Y_0 at the new observation point $\mathbf{X_0}$ is:

$$\left[\mathbf{X_0}'\widehat{\boldsymbol{\beta}} - t_{n-(p+1),0.975}\hat{\sigma}\sqrt{1 + \mathbf{X_0}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X_0}},\right.$$

$$\left.\mathbf{X_0}'\widehat{\boldsymbol{\beta}} + t_{n-(p+1),0.975}\hat{\sigma}\sqrt{1 + \mathbf{X_0}'(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X_0}}\right].$$

Exercise. — Recover the construction of the prediction intervals. Hint: what is the distribution of $\widehat{Y}_0 - Y_0$?

On the example of the Ozone data, with the - simple linear regression model with the single variable X= **MOCAGE**

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i,\ i = 1, \ldots, n,$$

we obtain the following confidence and prediction intervals.

2.4 Fisher test of a submodel

Suppose that our data obey to a polynomial regression model of degree p and we want to test the null hypothesis that our data obey to a polynomial regression model of degree $k < p$, hence we want to test that the $p - k$ last coefficients of $\boldsymbol{\beta}$ are equal to 0. More generally, assume that our data obey to the model, called Model (1):

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

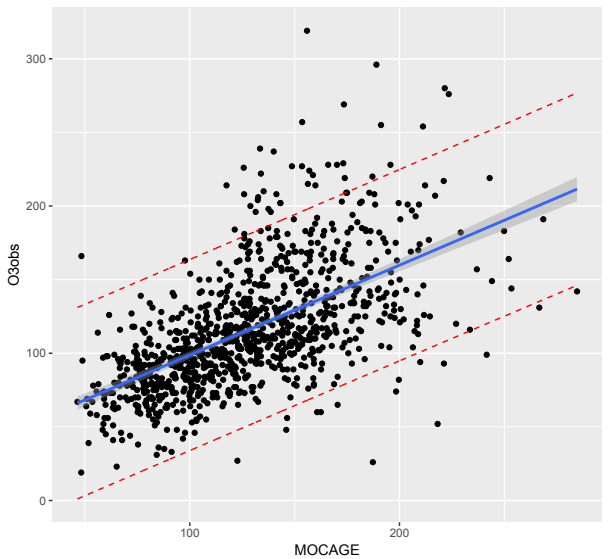


Figure 3.1: Simple linear regression model: confidence intervals for the mean response (in grey) and prediction intervals (red dotted lines)

where $\beta \in R^p$ and consider another model, called Model (0):

$$\mathbf{Y} = \tilde{\mathbf{X}}\boldsymbol{\theta} + \boldsymbol{\varepsilon}.$$

where $\theta \in \mathbb{R}^l$ with $l < p$.

DEFINITION 7. — *We define*

$$V = \{\mathbf{X}\boldsymbol{\beta}, \boldsymbol{\beta} \in \mathbb{R}^p\}$$

and

$$W = \{\tilde{\mathbf{X}}\boldsymbol{\theta}, \boldsymbol{\theta} \in \mathbb{R}^l\}.$$

We say that Model (0) is a submodel of Model (1) if W is a linear subspace of V.

We want to test the hypothesis:
H_0: "the vector $\mathbf{Y}$ of observations obeys to Model (0)" against the alternative
H_1: "the vector $\mathbf{Y}$ of observations obeys to Model (1)".
In the Model (0), the least square estimator of $\boldsymbol{\theta}$ is:

$$\widehat{\boldsymbol{\theta}} = \begin{pmatrix} \widehat{\boldsymbol{\theta}}_0 \\ \widehat{\boldsymbol{\theta}}_1 \\ . \\ . \\ \widehat{\boldsymbol{\theta}}_l \end{pmatrix} = (\tilde{\mathbf{X}}'\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}'\mathbf{Y}.$$

The F-statistics is defined by:

$$F = \frac{\|\mathbf{X}\widehat{\boldsymbol{\beta}} - \tilde{\mathbf{X}}\widehat{\boldsymbol{\theta}}\|^2/(p-l)}{\|\mathbf{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}}\|^2/(n-p)}.$$

An alternative way to write the F-statistics is:

$$F = \frac{(\mathrm{SSR}_{\mathbf{0}} - \mathrm{SSR}_{\mathbf{1}})/(p-l)}{\mathrm{SSR}_{\mathbf{1}}/(n-p)},$$

where SSR_0 and SSR_1 respectively denote the residuals sum of square under Model (0) and Model (1).

Exercise. — Prove that, under the null hypothesis H_0, the F-statistics is a Fisher distribution with parameters $(p-l, n-p)$.

The numerator of the F-statistics corresponds to $\left\|\widehat{\mathbf{Y}}_{\mathbf{0}} - \widehat{\mathbf{Y}}_{\mathbf{1}}\right\|^2$, where $\widehat{\mathbf{Y}}_{\mathbf{0}}$ and $\widehat{\mathbf{Y}}_{\mathbf{1}}$ correspond respectively to the predicted values under the sub-model and under the full model. This quantity is small under the null hypothesis, when the sub-model is valid, and becomes larger under the alternative. Hence, the null hypothesis is rejected for large values of F, namely, for a level-α test, when

$$F > f_{p-l,n-p,1-\alpha},$$

where $f_{p,q,1-\alpha}$ is the $(1-\alpha)$ quantile of the Fisher distribution with parameters (p,q). The statistical softwares provide the $p-$ value of the test:

$$P_{H_0}(F > F_{obs})$$

where F_{obs} is the observed value for the F-statistics. The null hypothesis is rejected at level α if the $p-$ value is smaller than α.

2.5 Diagnosis on the residuals

As illustrated for Ozone data on Figure 3.2, the analysis and visualisation of the residuals allow to verify some hypotheses:

- Homoscedasticity: the variance σ^2 is assumed to be constant,

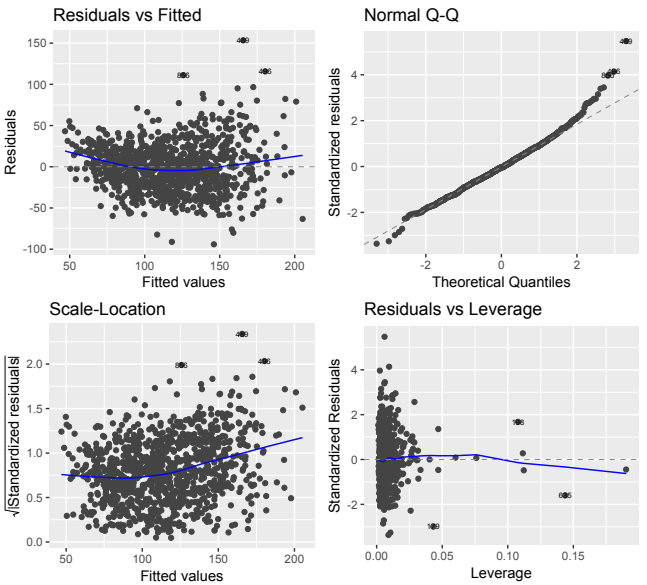


Figure 3.2: Diagnosis on the residuals for Ozone data

- The linear model is valid: there is no tendancy in the residuals,
- Detection of possible outliers with the Cook's distance
- Normality of the residuals (if this assumption was used to provide confidence/prediction intervals or tests).

This is rather classical for linear regression, and we focus here on the detection of possible high collinearities between the regressors, since it has an im-

pact on the variance of our estimators. Indeed, we have seen that the variance-covariance matrix of $\widehat{\boldsymbol{\beta}}$ is $\sigma^2(\mathbf{X}'\mathbf{X})^{-1}$.

When the matrix $\mathbf{X}$ is *ill-conditioned*, which means that the determinant of $\mathbf{X}'\mathbf{X}$ is close to 0, we will have high variances for some components of $\widehat{\boldsymbol{\beta}}$. It is therefore important to detect and remedy these situations by removing some variables of the model or introducing some constraints on the parameters to reduce the variance of the estimators.

VIF

Most statistical softwares propose collinearity diagnosis. The most classical il the *Variance Influence Factor* (VIF)

$$V_j = \frac{1}{1 - R_j^2}$$

where R_j^2 corresponds to the determination coefficient of the regression of the variable $\boldsymbol{X}^j$ on the other explanatory variables ; R_j represents also the cosine of the angle in $\mathbb{R}^n$ between $\boldsymbol{X}^j$ and the linear subspace generated by the variables $\{\boldsymbol{X}^1, \ldots, \boldsymbol{X}^{j-1}, \boldsymbol{X}^{j+1}, \ldots, \boldsymbol{X}^p\}$. The more $\boldsymbol{X}^j$ is "linearly" linked with the other variables, the more R_j is close to 1 ; we show that the variance of the estimator of β_j is large in this case. This variance is minimal when $\boldsymbol{X}^j$ is orthogonal to the subspace generated by the other variables.

Condition number

We consider the covariance matrix $\mathbf{R}$ between the regressors. We denote $\lambda_1 \geq \ldots \geq \lambda_p$ the ordered eigenvalues of $\mathbf{R}$. If the smallest eigenvalues are close to 0, the inversion of the matrix $\mathbf{R}$ will be difficult and numerical problems arise. In this case, some components of the least square estimator $\widehat{\boldsymbol{\beta}}$ will have high variances. The condition number of the matrix $\mathbf{R}$ is defined as the ratio

$$\kappa = \lambda_1/\lambda_p$$

between the largest and the smallest eigenvalues of $\mathbf{R}$. If this ratio is large, then the problem is ill-conditioned.
This condition number is a global indicator of collinearities, while the VIF allows to identify the variables that are problematic.

3 Determination coefficient and Model selection

3.1 R^2 and adjusted R^2

We define respectively the total, explicated and residual sums of squares by

$$\text{SST} = \sum_{i=1}^{n} (Y_i - \bar{Y})^2 = \left\| \mathbf{Y} - \overline{\mathbf{Y}}\mathbf{1} \right\|^2,$$

$$\text{SSE} = \sum_{i=1}^{n} (\hat{Y}_i - \bar{Y})^2 = \left\| \widehat{\mathbf{Y}} - \overline{\mathbf{Y}}\mathbf{1} \right\|^2,$$

$$\text{SSR} = \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 = \left\| \mathbf{Y} - \widehat{\mathbf{Y}} \right\|^2 = \|\mathbf{e}\|^2.$$

Since we consider a model with intercept, by Pythagora's theorem,

$$\left\| \mathbf{Y} - \overline{\mathbf{Y}}\mathbf{1} \right\|^2 = \left\| \mathbf{Y} - \widehat{\mathbf{Y}} \right\|^2 + \left\| \widehat{\mathbf{Y}} - \overline{\mathbf{Y}}\mathbf{1} \right\|^2,$$

we have the following identity:

$$\text{SST} = \text{SSR} + \text{SSE}.$$

We define the determination coefficient R^2 by:

$$R^2 = \frac{\text{SSE}}{\text{SST}} = 1 - \frac{\text{SSR}}{\text{SST}}.$$

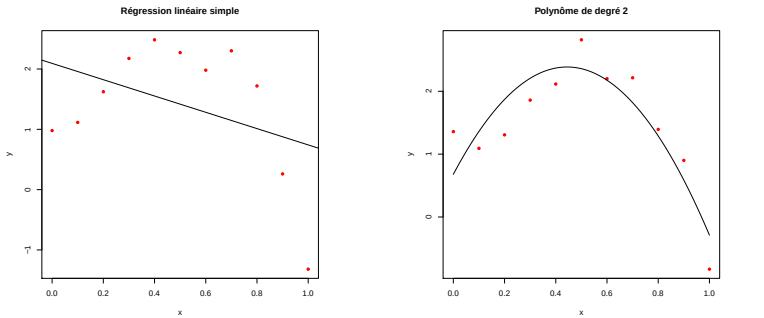


Figure 3.3: Polynomial regression: adjusted model, on the left: $y = \beta_0 + \beta_1 x + \epsilon$, $R^2 = 0.03$, on the right: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \epsilon$, $R^2 = 0.73$.

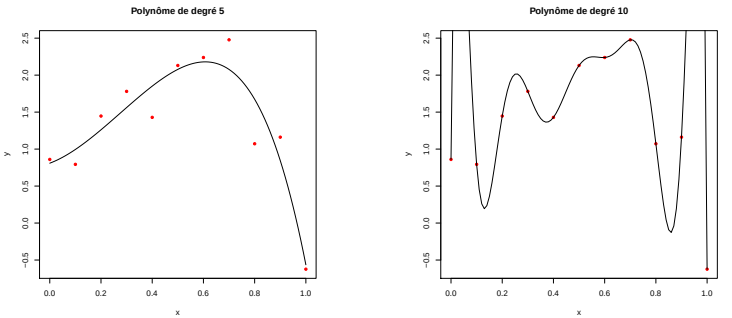


Figure 3.4: Polynomial regression: adjusted model, on the left: $y = \beta_0 + \beta_1 x + \ldots + \beta_5 x^5 + \epsilon$, $R^2 = 0.874$, on the right: $y = \beta_0 + \beta_1 x + \ldots + \beta_{10} x^{10} + \epsilon$, $R^2 = 1$.

Note that $0 \leq R^2 \leq 1$. The model is well adjusted to the n training data if the residuals sum of square SSR is close to 0, or equivalently, if the determination coefficient R^2 is close to 1. Hence, the first hint is that a "good" model is a model for which R^2 is close to 1. This is in fact not true, as shown by the following pedagogical example of polynomial regression. Suppose that we have a training sample $(X_i, Y_i)_{1 \leq i \leq n}$ where $X_i \in [0, 1]$ and $Y_i \in \mathbb{R}$ and we adjust polynomials on these data:

$$Y_i = \boldsymbol{\beta}_0 + \boldsymbol{\beta}_1 X_i + \boldsymbol{\beta}_2 X_i^2 + \ldots + \boldsymbol{\beta}_k X_i^k + \varepsilon_i.$$

When k increases, the model is more and more complex, hence $\left\|\mathbf{Y} - \widehat{\mathbf{Y}}\right\|^2$ decreases, and R^2 increases as shown in Figures 3.3 and 3.4.

The determination coefficient is equal to 1 for the polynomial of degree $n-1$ (which has n coefficients) and passes through all the training points. Of course this model is not the best one: it has a very high variance since we estimate as much coefficients as the number of observations. This is a typical case of *overfitting*. When the degree of the polynomial increases, the bias of our estimators decreases, but the variance increases. The best model is the one that realizes the best trade-off between the bias term and the variance term. Hence, we have seen that maximizing the determination coefficient is not a good criterion to compare models with various complexity. It is more interesting to consider the adjusted determination coefficient defined by:

$$R'^2 = 1 - \frac{\text{SSR}/(n-k-1)}{\text{SST}/(n-1)}.$$

The definition of R'^2 takes into account the complexity of the model, represented here by its number of coefficients: $k+1$ for a polynomial of degree k, and penalizes more complex models. One can choose, between several mod-

els, the one which maximizes the adjusted R^2. In the previous example, we would choose a polynomial of degree 3 with this criterion.
More generally, we have to define model selection procedures that realize a good compromise between a good adjustment to the data (small bias) and a small variance; and an unbiased estimator is not necessarily the best one in this sense. We will prefer a biased model if this allows to reduce drastically the variance. There are several ways to do that:

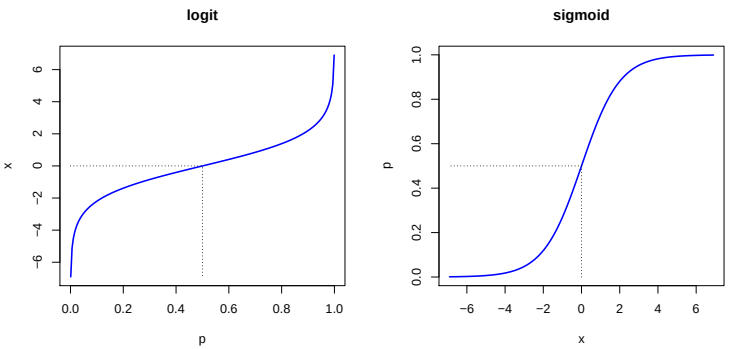
- Reducing the number of explanatory variables and by the same way simplifying the model (variable selection or *Lasso* penalization)
- Putting some constraints on the parameters of the model by *shrinking* them (*Ridge* or *Lasso* penalization)

This penalized criterion will be the topic of the Chapter 4.

4 Logistic regression

We assume that $\mathcal{X} = \mathbb{R}^p$. One of the most popular model for binary classification when $\mathcal{Y} = \{-1, 1\}$ is the **logistic regression** model. The idea for logistic regression is to use a linear model for probabilities, thanks to a one-to-one mapping ("link" function) from $[0, 1]$ to $\mathbb{R}$.
The most used is the **logit** function and its inverse, the **sigmoid** function:

	$[0,1]$		$\mathbb{R}$	
logit:	π	$\rightarrow$	$\ln\left(\frac{\pi}{1-\pi}\right)$	
	$\frac{\exp(x)}{1+\exp(x)}$	$\leftarrow$	x	: **sigmoid**



Other link functions can be considered such as :

- The **probit** function $g(\pi) = F^{-1}(\pi)$ where F is the distribution function of the standard normal distribution.
- The **log-log** function $g(\pi) = \ln(-\ln(1-\pi))$.

4.1 The model

This leads to the following formulation for the logistic regression model:

$$\pi_{\boldsymbol{\beta}}(\boldsymbol{x}) = \mathbb{P}_{\boldsymbol{\beta}}(Y = 1/\boldsymbol{X} = \boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle)}{1 + \exp(\langle \boldsymbol{\beta}, \boldsymbol{x} \rangle)} \text{ for all } x \in \mathcal{X},$$

with $\boldsymbol{\beta} \in \mathbb{R}^p$.

Exercise. — Compute the Bayes classifier f^* for this model and determine the border between $f^* = 1$ and $f^* = -1$.

4.2 Estimation of the parameters

Given a n-sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$, we can estimate the parameter $\boldsymbol{\beta}$ by maximizing the conditional likelihood of $\underline{Y} = (Y_1, \ldots, Y_n)$ given $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_n)$. Since the distribution of Y given $\boldsymbol{X} = \boldsymbol{x}$ is a Bernoulli distribution with parameter $\pi_{\boldsymbol{\beta}}(\boldsymbol{x})$, the conditional likelihood is

$$L(Y_1, \ldots, Y_n, \boldsymbol{\beta}) = \prod_{i=1}^{n} \pi_{\boldsymbol{\beta}}(\boldsymbol{X_i})^{Y_i} (1 - \pi_{\boldsymbol{\beta}}(\boldsymbol{X_i}))^{1-Y_i}.$$

$$L(\underline{Y}, \boldsymbol{\beta}) = \prod_{i, Y_i=1} \frac{\exp(\langle \boldsymbol{\beta}, \boldsymbol{X_i} \rangle)}{1 + \exp(\langle \boldsymbol{\beta}, \boldsymbol{X_i} \rangle)} \prod_{i, Y_i=0} \frac{1}{1 + \exp(\langle \boldsymbol{\beta}, \boldsymbol{X_i} \rangle)}.$$

- Unlike the linear model, there is no explicit expression for the maximum likelihood estimator $\hat{\boldsymbol{\beta}}$.
- It can be shown that computing $\hat{\boldsymbol{\beta}}$ is a convex optimization problem.
- We compute the gradient of the log-likelihood, also called **the score function** $S(\underline{Y}, \beta)$ and use a **Newton-Raphson algorithm** to approximate $\hat{\boldsymbol{\beta}}$ satisfying $S(\underline{Y}, \hat{\boldsymbol{\beta}}) = 0$.

We then compute the logistic regression classifier:

$$\forall x \in \mathcal{X}, \hat{f}(\boldsymbol{x}) = \text{sign}(\langle \hat{\boldsymbol{\beta}}, \boldsymbol{x} \rangle).$$

An illustration of the logistic regression for one-dimensional predictors in presented in Figure 3.5.

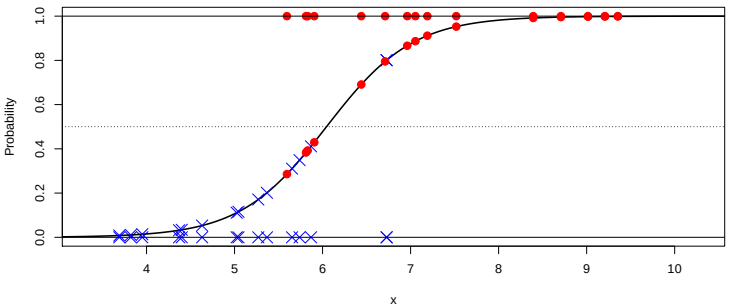


Figure 3.5: Logistic regression for a dataset composed of 2 groups of size 15, sampled from Normal distributions, centered at 5 and 7, with variance 1.

Like for linear models, in a high dimensional setting (p is large), it will be necessary to use variable selection and model selection procedures by introducing penalized likelihood criterions (AIC, BIC, LASSO ..). This is the topic of Chapter 4.

Chapter 4

Model selection for linear models

1 Introduction

We have made some reminders on linear models in Chapter 3. We have seen that, in a high dimensional framework, when p is possibly large, even larger than n, a complete model obtained by least square estimation is overfitted and it is necessary to regularize the least square estimation by introducing some penalty on the complexity of the models in order to reduce the variance of the estimators. The adjusted R^2, presented in Chapter 3 is a first step in this direction. Model selection and variable selection for linear models has been intensively studied this past twenty years and is still a very active field of research in statistics. Some of these methods such as Ridge or Lasso methods, will be at the core of this course.

2 Variable selection

As we have seen, the least square estimator is not satisfactory since it has low bias but generally high variance. In most examples, several variables are not significant, and we may have better results by removing those variables from the model. Moreover, a model with a small number of variables is more interesting for the interpretation, keeping only the variables that have the strongest effects on the variable to explain. There are several ways to do that.

Assume we want to select a subset of variables among all possible subsets taken from the input variables. Each subset defines a model, and we want to select the "best model". We have seen that maximizing the R^2 is not a good criterion since this will always lead to select the full model. It is more interesting to select the model maximizing the adjusted determination coefficient R'^2. Many other penalized criterion have been introduce for variable selection such as the Mallow's C_P criterion or the BIC criterion. In both cases, it corresponds to the minimization of the least square criterion plus some penalty term, depending on the number k of parameters in the model m that is considered.

$$\text{Crit}(m) = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 + \text{pen(k)}.$$

The Mallow's C_P criterion is

$$\text{Crit}_{C_P}(m) = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + 2k\sigma^2,$$

and the BIC criterion penalizes more the dimension of the model with an additional logarithmic term.

$$\text{Crit}_{BIC}(m) = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 + \log(n)k\sigma^2.$$

The aim is to select the model (among all possible subsets) that minimizes one of those criterion. On the example of the polynomial models, we obtain the results summarized in Figure 4.1.

Nevertheless, the number of subsets of a set of p variables is 2^p, and it is impossible (as soon as $p > 30$) to explore all the models to minimize the criterion. Fast algorithms have been developed to find a clever way to explore a subsample of the models. This are the *backward, forward* and *stepwise* algorithms.

Backward/Forward Algorithms:

- **Forward selection:** We start from the constant model (only the intercept, no explanatory variable), and we add sequentially the variable that allows to reduce the more the criterion.
- **Backward selection:** This is the same principle, but starting from the full model and removing one variable at each step in order to reduce the criterion.
- **Stepwise selection:** This is a mixed algorithm, adding or removing one variable at each step in order to reduce the criterion in the best way.

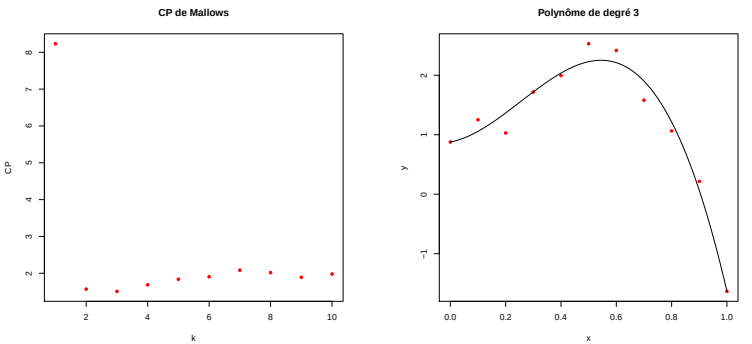


Figure 4.1: Mallows'C_P in function of the degree of the polynomial. Selected model: polynomial with degree 3.

All those algorithms stop when the criterion can no more be reduced. Let us see some applications of those algorithms on the Ozone data.

Stepwise Algorithm

We apply the `StepAIC` algorithm, with the option **both** of the software R in order to select a subset of variables, and we present here an intermediate result:

```
                    Start:  AIC=6953.05
O3obs ~ MOCAGE + TEMPE + RMH2O + NO2 + NO + VentMOD +
    VentANG
```

```
            Df  Sum of Sq  RSS      AIC
- VentMOD   1   1484       817158   6952.9
<none>                     815674   6953.0
- RMH2O     1   4562       8202354  6956.9
- VentANG   1   12115      827788   6966.4
- NO2       1   21348      837022   6977.9
- NO        1   21504      837178   6978.1
- MOCAGE    1   225453     1041127  7205.1
- TEMPE     1   268977     1084651  7247.7
            Step:  AIC= 6952.94
O3obs ~ MOCAGE + TEMPE + RMH2O + NO2 + NO + VentANG
```

3 Ridge regression

The principle of the Ridge regression is to consider all the explanatory variables, but to introduce constraints on the parameters in order to avoid overfitting, and by the same way in order to reduce the variance of the estimators. In the case of the Ridge regression, we introduce an l_2 constraint on the parameter $\boldsymbol{\beta}$.

3.1 Model and estimation

If we have an ill-conditionned problem, but we want to keep all the variables, it is possible to improve the numerical properties and to reduce the variance of the estimator by considering a slightly biased estimator of the parameter $\boldsymbol{\beta}$.

We consider the linear model

$$\mathbf{Y} = \widetilde{\mathbf{X}}\widetilde{\boldsymbol{\beta}} + \boldsymbol{\epsilon},$$

where

$$\widetilde{\mathbf{X}} = \begin{pmatrix} 1 & X_1^1 & X_1^2 & . & X_1^p \\ 1 & X_2^1 & X_2^2 & . & X_2^p \\ . & . & . & . & . \\ 1 & X_n^1 & X_n^2 & . & X_n^p \end{pmatrix},$$

$$\widetilde{\boldsymbol{\beta}} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ . \\ . \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ \beta_p \end{pmatrix}.$$

We set $\boldsymbol{X}^0 = (1, 1, \ldots, 1)'$, and $\mathbf{X}$ the matrix $\widetilde{\mathbf{X}}$ where we have removed the first column. The *ridge* estimator is defined by a least square criterion plus a penalty term, with an l_2 type penalty.

DEFINITION 8. — *The* ridge *estimator of $\widetilde{\boldsymbol{\beta}}$ in the model*

$$\mathbf{Y} = \widetilde{\mathbf{X}}\widetilde{\boldsymbol{\beta}} + \boldsymbol{\epsilon},$$

is defined by

$$\widehat{\boldsymbol{\beta}} = \textit{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left(\sum_{i=1}^{n} (Y_i - \sum_{j=0}^{p} X_i^{(j)} \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right),$$

where λ is a non negative parameter, that we have to calibrate.

Note that the parameter β_0 is not penalized.

PROPOSITION 1. — *Assume that $\boldsymbol{X}$ is centered. We obtain the following ex-*

plicit solution for the Ridge estimator:

$$\hat{\beta}_0 = \bar{Y},\ \hat{\beta}_R = \begin{pmatrix} \hat{\beta}_1 \\ \cdot \\ \cdot \\ \hat{\beta}_p \end{pmatrix} = (\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_p)^{-1}\mathbf{X}'(\mathbf{Y} - \bar{Y}\mathbf{1}).$$

Exercise. — Prove the Proposition 1.

Remarks:

1. $\mathbf{X}'\mathbf{X}$ is a nonnegative symmetric matrix (for all vector $\boldsymbol{u}$ in $\mathbb{R}^p$, $\boldsymbol{u}'(\mathbf{X}'\mathbf{X})\boldsymbol{u} = \|\mathbf{X}\boldsymbol{u}\|^2 \geq 0$. Hence, for any $\lambda > 0$, $\mathbf{X}'\mathbf{X} + \lambda \mathbf{I}_p$ is invertible.

2. The constant β_0 is not penalized, otherwise, the estimator would depend on the choice of the origin for $\mathbf{Y}$. We obtain $\widehat{\beta}_0 = \overline{\mathbf{Y}}$, adding a constant to $\mathbf{Y}$ does not modify the values of $\widehat{\beta}_j$ for $j \geq 1$.

3. The *ridge* estimator is not invariant by normalization of the vectors $X^{(j)}$, it is therefore important to normalize the vectors before minimizing the criterion.

4. The *ridge* regression is equivalent to the least square estimation under the constraint that the l_2-norm of the vector $\boldsymbol{\beta}$ is not too large:

$$\widehat{\boldsymbol{\beta}}_R = \arg\min_{\beta} \left\{ \|\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}\|^2 ; \|\boldsymbol{\beta}\|^2 < c \right\}.$$

The ridge regression keeps all the parameters, but, introducing constraints on the values of the β_j's avoids too large values for the estimated parameters, which reduces the variance.

Choice of the penalty term

In the Figure 4.2, we see results obtained by the *ridge* method for several values of the tuning parameter $\lambda = l$ on the polynomial regression example. Increasing the penalty leads to more regular solutions, the bias increases, and the variance decreases. We have overfitting when the penalty is equal to 0 and under-fitting when the penalty is too large.

For each regularization method, the choice of the parameter λ is crucial and determinant for the model selection. We see in Figure 4.3 the *Regularisation path*, showing the profiles of the estimated parameters when the tuning parameter λ increases.

Choice of the regularization parameter

Most softwares use the cross-validation to select the tuning parameter penalty. The principe is the following:

- We split the data into K sub-samples. For all I from 1 to K:
  - We compute the Ridge estimator associated to a regularization parameter λ from the data of all the subsamples, except the I-th (that will be a "'test"' sample).
  - We denote by $\hat{\boldsymbol{\beta}}_\lambda^{(-I)}$ the obtained estimator.
  - We test the performances of this estimator on the data that have not been used to build it, that is the one of the I-th sub-sample.
- We compute the criterion:

$$CV(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{Y}_i - \boldsymbol{X}_i\hat{\boldsymbol{\beta}}_\lambda^{(-\tau(i))})^2.$$

- We choose the value of λ which minimizes $CV(\lambda)$.

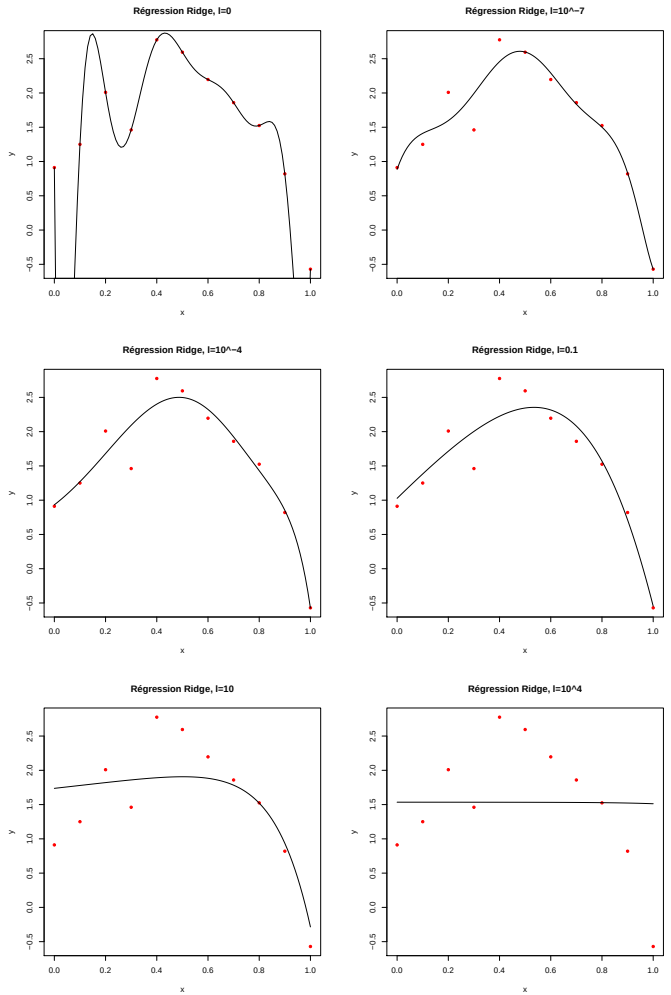


Figure 4.2: *Ridge* penalisation for the polynomial model

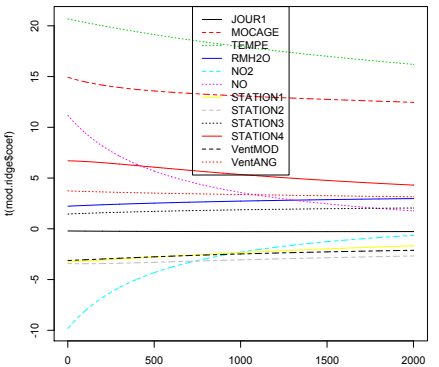


Figure 4.3: Regularization paths for the Ridge regression

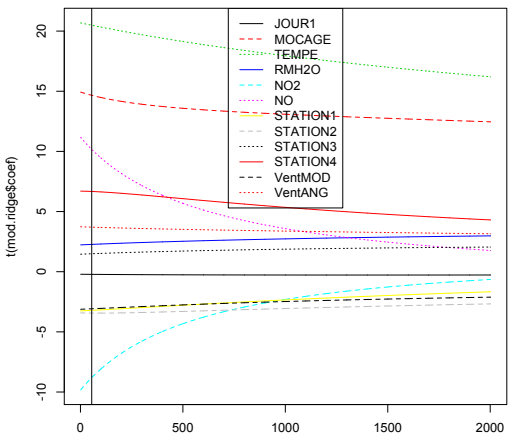


Figure 4.4: Selection of the regularization parameter by CV

Application to the Ozoner data: The value of λ selected by cross-validation is 5.4. We show the obtained value in Figure 4.4.

Singular Value Decomposition and Ridge regression

The Singular Value Decomposition (SVD) of the centered matrix $\mathbf{X}$ allows to interpret the *ridge* regression as a shrinkage method. The SVD of the matrix $\boldsymbol{X}$ has the following form:

$$\mathbf{X} = \mathbf{UDV}',$$

where $\mathbf{X}$ is a $n \times p$ matrix, $\mathbf{U}$ is $n \times n$, $\mathbf{D}$ is a $n \times p$ "diagonal" matrix whose all elements are ≥ 0 and ordered by decreasing values, $\mathbf{V}$ is a $p \times p$ matrix. The elements of D are the singular values of the matrix X. $\mathbf{U}$ and $\mathbf{V}$ are orthogonal: $\mathbf{UU}' = \mathbf{U}'\mathbf{U} = \mathbf{I}_n$, $\mathbf{VV}' = \mathbf{V}'\mathbf{V} = \mathbf{I}_p$.
We have

$$\mathbf{X}\widehat{\boldsymbol{\beta}}_R = \mathbf{UD}(\mathbf{D}'\mathbf{D} + \lambda \mathbf{I}_p)^{-1}\mathbf{D}'\mathbf{U}'\mathbf{Y}.$$

Suppose that $n \leq p$. We denote by $\boldsymbol{u}^{(1)}, \ldots, \boldsymbol{u}^{(n)}$ the columns of the matrix $\mathbf{U}$. Setting $d_1 \geq \ldots \geq d_p \geq 0$ the diagonal elements of $\mathbf{D}$, $\mathbf{UD}$ is a $n \times p$ matrix whose j-th column is $d_j \boldsymbol{u}^{(j)}$. We therefore have

$$\mathbf{X}\widehat{\boldsymbol{\beta}}_R = \sum_{j=1}^{p} \boldsymbol{u}^{\boldsymbol{j}} \left(\frac{d_j^2}{d_j^2 + \lambda} \right) (\boldsymbol{u}^{\boldsymbol{j}})'\mathbf{Y}.$$

Let us compare this estimator with the least square estimator (which corresponds to $\lambda = 0$):

$$X\widehat{\boldsymbol{\beta}} = \sum_{j=1}^{p} \boldsymbol{u}^{\boldsymbol{j}} (\boldsymbol{u}^{\boldsymbol{j}})'\mathbf{Y}.$$

$(\boldsymbol{u}^{\boldsymbol{j}})'\mathbf{Y}$ corresponds to the j-th component of $\mathbf{Y}$ in the basis $(\boldsymbol{u}^{\boldsymbol{1}}, \ldots, \boldsymbol{u}^{\boldsymbol{n}})$.
In the case of the *ridge* regression, this component is multiplied by the factor $d_j^2/\left(d_j^2 + \lambda\right) \in]0, 1[$, we can say that this component has been *thresholded*.
Remarks:
1) When the tuning parameter λ increases, the coefficients are more and more thresholded.
2) $x \mapsto x/(x + \lambda)$ is a non decreasing function of x for $x > 0$. The largest coefficients are slightly thresholded: if $d_j^2 >> \lambda$, $d_j^2/\left(d_j^2 + \lambda\right)$ is close to 1. The threshold decreases when j increases since d_j decreases.

We can give an interpretation in relation with the **Principal Components Analysis** . $\mathbf{X}$ being centered, $\mathbf{X}'\mathbf{X}/n$ is the empirical variance-covariance

matrix of the column vectors of the matrix $\boldsymbol{X}$.

$$\mathbf{X}'\mathbf{X} = \mathbf{V}\mathbf{D}'\mathbf{D}\mathbf{V}',$$

where $\mathbf{D}'\mathbf{D}$ is the diagonal matrix composed by the elements d_i^2. We denote by $\boldsymbol{v_1}, \ldots, \boldsymbol{v_p}$ the column vectors in $\mathbb{R}^p$ of the matrix $\mathbf{V}$.
Let $\boldsymbol{v}$ be an $\mathbb{R}^p$ vector with norm 1.

$$\hat{Var}(\mathbf{X}\mathbf{v}) = \frac{1}{n}(\mathbf{X}\mathbf{v})'(\mathbf{X}\mathbf{v}) = \frac{1}{n}\boldsymbol{v}'(\mathbf{X}'\mathbf{X})\boldsymbol{v},$$

which is maximal for $\boldsymbol{v} = \boldsymbol{v_1}$ and is equal to d_1^2.
$\boldsymbol{z}_1 = \mathbf{X}\boldsymbol{v_1}$ is the first principal component of the matrix $\mathbf{X}$.
The orthonormal eigenvectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_p$ are the principal directions (or Karhunen Loeve directions) of $\mathbf{X}$. The variables $\boldsymbol{z}_j = X\boldsymbol{v}_j$ are the principal components. We remark that

$$\boldsymbol{z}_j = X\boldsymbol{v}_j = \mathbf{U}\mathbf{D}\mathbf{V}'\boldsymbol{v}_j = d_j\boldsymbol{u^{(j)}}.$$

We see that the *ridge* regression shrinks slightly the first principal components (for which d_j is large), and more the last principal components.
We can associate to the *ridge* procedure the quantity $df(\lambda)$ which is called the effective number of degrees of freedom in the *ridge* regression and is defined by

$$df(\lambda) = \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda}.$$

If $\lambda = 0$, $df(\lambda) = p$ (no shrinkage), if $\lambda \to \infty$, $df(\lambda) \to 0$, at the limit, all the coefficients are equal to 0.

4 The LASSO regression

The *ridge* regression allows to get around the collinearity problems even if the numbers of predictors p is large with possibly $p > n$. The main weakness of this method is related to interpretation difficulties because, without selection, all variables are included in the model. Other regularization approaches also allow selection, as the LASSO regression, which leads to more interpretable solutions.

4.1 Model and estimation

LASSO is the abbreviation of Least Absolute Shrinkage and Selection Operator. The Lasso estimator is introduced in the paper by Tibshirani, R. (1996)[38]: Regression shrinkage and selection via the lasso. J. Royal. Statist. Soc B., Vol. 58, No. 1, pages 267-288. The Lasso corresponds to the minimization of a least square criterion plus an l_1 penalty term (and no more an l_2 penalization like in the *ridge* regression). We denote $\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$.

DEFINITION 9. — *The Lasso estimator of $\boldsymbol{\beta}$ in the model*

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

is defined by:

$$\widehat{\boldsymbol{\beta}}_{Lasso} = argmin_{\boldsymbol{\beta}\in\mathbb{R}^{p+1}} \left(\sum_{i=1}^{n} (Y_i - \sum_{j=0}^{p} X_i^{(j)}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right),$$

where λ is a nonnegative tuning parameter.

We can show that this is equivalent to the minimization problem:

$$\widehat{\boldsymbol{\beta}}_L = \text{argmin}_{\boldsymbol{\beta}\in\mathbb{R}^p, \|\boldsymbol{\beta}\|_1 \leq t}(\|\mathbf{Y} - \mathbf{X}\beta\|^2),$$

where t is suitably chosen, and $\widehat{\beta_0}_{\text{Lasso}} = \bar{Y}$. Like for the Ridge regression, the parameter λ is a regularization parameter:

- If $\lambda = 0$, we recover the least square estimator.
- If λ tends to infinity, all the coefficients $\hat{\beta}_j$ are equal to 0 for $j = 1, \ldots, p$.

The solution to the Lasso is parsimonious (or sparse), since it has many null coefficients.

If the matrix $\mathbf{X}$ is orthogonal: ($\mathbf{X}'\mathbf{X} = Id$), the solution is explicit.

PROPOSITION 2. — *If $\mathbf{X}'\mathbf{X} = \mathbf{I}_p$, the solution $\boldsymbol{\beta}$ of the minimization of the Lasso criterion*

$$\|\mathbf{Y} - \mathbf{X}\beta\|^2 + 2\lambda\|\boldsymbol{\beta}\|_1$$

is defined as follows: for all $j = 1, \ldots, p$,

$$\beta_j = sign(\widehat{\beta}_j)(|\widehat{\beta}_j| - \lambda)\mathbf{1}_{|\widehat{\beta}_j| \geq \lambda},$$

where $\widehat{\boldsymbol{\beta}}$ is the least square estimator: $\widehat{\boldsymbol{\beta}} = \mathbf{X}'\mathbf{Y}$.

The obtained estimator corresponds to a soft thresholding of the least square estimator. The coefficients $\widehat{\beta}_j$ are replaced by $\phi_\lambda(\widehat{\beta}_j)$ where

$$\phi_\lambda : x \mapsto \text{sign}(x)(|x| - \lambda)_+.$$

Exercise. — Prove the proposition 2.

Another formulation

The LASSO is equivalent to the minimization of the criterion

$$\text{Crit}(\beta) = \sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i^{(1)} - \beta_2 X_i^{(2)} - \ldots - \beta_p X_i^{(p)})^2$$

under the constraint $\sum_{j=1}^{p} |\beta_j| \leq t$, for some $t > 0$ (depending on λ).

The statistical software R introduces a constraint expressed by a relative bound for $\sum_{j=1}^{p} |\beta_j|$: the constraint is expressed by

$$\sum_{j=1}^{p} |\beta_j| \leq \kappa \sum_{j=1}^{p} |\hat{\beta}_j^{(0)}|,$$

where $\hat{\beta}^{(0)}$ is the least square estimator and $\kappa \in [0, 1]$.

For $\kappa = 1$ we recover the least square estimator (there is no constraint) and for $\kappa = 0$, all the $\hat{\beta}_j$, $j \geq 1$, vanish (maximal constraint).

4.2 Applications

We represent in Figure 4.5 the values of the coefficients in function of κ for the Ozone data: this are the regularization paths of the LASSO. As for the Ridge regression, the tuning parameter is generally calibrated by cross-validation.

Comparison LASSO/ RIDGE

The Figure 4.6 gives a geometric interpretation of the minimization problems for both the Ridge and Lasso estimators. This explains why the Lasso solution is sparse.

4.3 Optimization algorithms for the LASSO

Convex functions and subgradients

DEFINITION 10. — *A function $F : \mathbb{R}^n \to \mathbb{R}$ is convex if $\forall x, y \in \mathbb{R}^n, \forall \lambda \in [0, 1]$,*

$$F(\lambda x + (1 - \lambda)y) \leq \lambda F(x) + (1 - \lambda)F(y).$$

LEMMA 3. — *When F is differentiable, we have $F(y) \geq F(x) + \langle \nabla F(x), y -$*

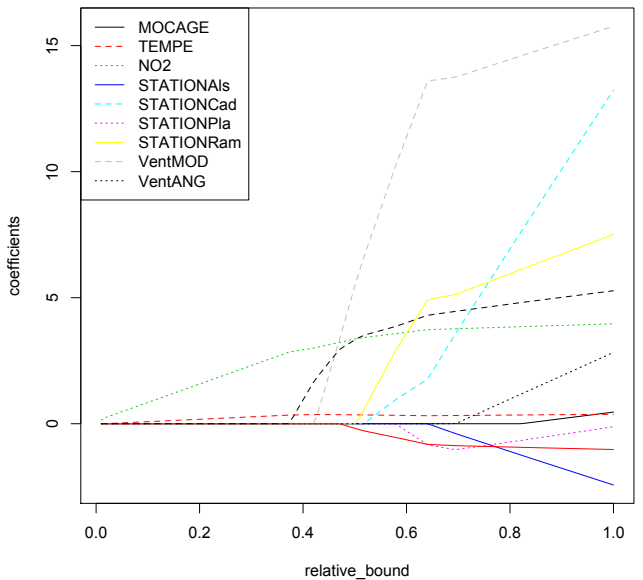


Figure 4.5: Regularization paths of the LASSO when the penalty decreases

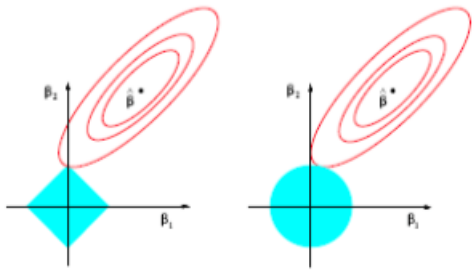


Figure 3.12: *Estimation picture for the lasso (left) and ridge regression (right). Shown are contours of the error and constraint functions. The solid blue areas are the constraint regions* $|\beta_1| + |\beta_2| \leq t$ *and* $\beta_1^2 + \beta_2^2 \leq t^2$*, respectively, while the red ellipses are the contours of the least squares error function.*

Figure 4.6: Geometric interpretation of Ridge and Lasso estimators

$x\rangle \ \forall y \in \mathbb{R}^n, \forall x \in \mathbb{R}^n$.

When F is non differentiable, we introduce the subdifferential ∂F of F defined by:

DEFINITION 11. — *The subdifferential ∂F of F is:*

$$\partial F(x) = \{\omega \in \mathbb{R}^n, F(y) \geq F(x) + \langle \omega, y - x\rangle, \forall y \in \mathbb{R}^n\}.$$

A vector $\omega \in \partial F(x)$ is called a subgradient of F in x.

LEMMA 4. — *F is convex $\Leftrightarrow \partial F(x) \neq \emptyset \ \forall x \in \mathbb{R}^n$.*

Example: subdifferential of the l_1 norm

$$\partial |x|_1 = \{\omega \in \mathbb{R}^n, \omega_j = 1 \text{ for } x_j > 0, \omega_j = -1 \text{ for } x_j < 0,$$

$$\omega_j \in [-1, 1] \text{ for } x_j = 0\}.$$

Remark: The subdifferential of a convex function is monotone in the following sense:

$$\langle \omega_x - \omega_y, x - y\rangle \geq 0 \ \forall \omega_x \in \partial F(x), \forall \omega_y \in \partial F(y).$$

Indeed

$$\begin{aligned} F(y) &\geq F(x) + \langle \omega_x, y - x\rangle \\ F(x) &\geq F(y) + \langle \omega_y, x - y\rangle. \end{aligned}$$

By summing, $\langle \omega_x - \omega_y, x - y\rangle \geq 0$.

First optimality condition

PROPOSITION 5. — *Let $F : \mathbb{R}^n \to \mathbb{R}$ be a convex function.*

$$x_* \in \textit{argmin}_{x \in \mathbb{R}^n} F(x) \Leftrightarrow 0 \in \partial F(x_*).$$

Proof: In both cases,

$$F(y) \geq F(x_*) + \langle 0, y - x_*\rangle.$$

The Lasso estimator

We consider the linear model:

$$Y = X\beta^* + \varepsilon.$$

We assume that the columns of X have norm 1. Let

$$L(\beta) = \|Y - X\beta\|^2 + \lambda |\beta|_1.$$

By definition, the Lasso estimator

$$\hat{\beta}_\lambda \in \text{argmin}_{\beta \in \mathbb{R}^p}(L(\beta)).$$

We deduce from the first order optimality condition that $0 \in \partial L(\hat{\beta}_\lambda)$.
We have that

$$L(\beta) = \|Y\|^2 - \beta' X' X\beta - 2\beta' XY + \lambda |\beta|_1.$$

LEMMA 6. — *Let $h : \beta \mapsto \beta' A\beta$ where A is a symmetric matrix. Then $\bigtriangledown h(\beta) = 2A\beta$.*

Let $g : \beta \mapsto \beta' z = z'\beta = \langle z, \beta\rangle$ where $z \in \mathbb{R}^p$. Then $\bigtriangledown g(\beta) = z$.

Hence we have

$$\partial L(\beta) = 2X'X\beta - 2X'Y + \lambda\partial|\beta|_1.$$

$$0 \in \partial L(\hat{\beta}_\lambda) \Leftrightarrow \exists \hat{z} \in \partial|\hat{\beta}_\lambda|_1 \text{ such that } :$$

$$2X'X\hat{\beta}_\lambda - 2X'Y + \lambda\hat{z} = 0.$$

This last equality is equivalent to

$$X'X\hat{\beta}_\lambda = X'Y - \frac{\lambda}{2}\hat{z} \qquad (E).$$

We have seen that

$$\begin{array}{lcl} \hat{z}_j & = & sign((\hat{\beta}_\lambda)_j \text{ if } (\hat{\beta}_\lambda)_j \neq 0 \\ \hat{z}_j & & \text{can be any real in } [-1,1] \text{ if } (\hat{\beta}_\lambda)_j = 0. \end{array}$$

Orthogonal setting

When $X'X = I_p$, (E) gives $(\hat{\beta}_\lambda)_j = X_j'Y - \frac{\lambda}{2}\hat{z}_j$.
Moreover, $\hat{z}_j = sign(\hat{\beta}_\lambda)_j$if $(\hat{\beta}_\lambda)_j \neq 0$. Hence,

$$\left\{ \begin{array}{l} (\hat{\beta}_\lambda)_j > 0 \Rightarrow X_j'Y > \frac{\lambda}{2} \\ (\hat{\beta}_\lambda)_j < 0 \Rightarrow X_j'Y < -\frac{\lambda}{2}. \end{array} \right. .$$

$$(\hat{\beta}_\lambda)_j \neq 0 \Rightarrow \left\{ \begin{array}{l} |X_j'Y| > \frac{\lambda}{2} \\ sign((\hat{\beta}_\lambda)_j) = sign(X_j'Y) \end{array} \right. .$$

This leads to the **explicit solution of the Lasso in the orthogonal setting**

$$(\hat{\beta}_\lambda)_j = sign(X_j'Y)\left(|X_j'Y| - \frac{\lambda}{2}\right)\mathbf{1}_{|X_j'Y|>\frac{\lambda}{2}}.$$

It corresponds to a **soft thresholding** of the Ordinary Least Square estimator $\hat{\beta}_j = X_j'Y$.

Non orthogonal setting

In this case, there is no analytic formula for the Lasso estimator $\hat{\beta}_\lambda$.
Let $\hat{m}_\lambda = \left\{j, (\hat{\beta}_\lambda)_j \neq 0\right\}$ be the support of $\hat{\beta}_\lambda$.
We can derive from Equation (E) that

- If $\lambda \geq 2\sup_j |X_j'Y|$, then $\hat{\beta}_\lambda = 0$.
- If $\lambda < 2\sup_j |X_j'Y|$, then denoting $X_{\hat{m}_\lambda}$ the submatrix obtained from X by keeping only the columns belonging to $\hat{m}_\lambda$, we have the following equation:

$$X'_{\hat{m}_\lambda}X_{\hat{m}_\lambda}(\hat{\beta}_\lambda)_{\hat{m}_\lambda} = X'_{\hat{m}_\lambda}Y - \frac{\lambda}{2}sign((\hat{\beta}_\lambda)_{\hat{m}_\lambda}).$$

Computing the Lasso estimator

$\beta \mapsto L(\beta) = \|Y - X\beta\|^2 + \lambda|\beta|_1$ is convex.
Hence a simple and efficient approach to minimize this function is to alternate minimization over each coordinate of β.
This algorithm converges to the Lasso estimator thanks to the convexity of L.
If we assume that the columns of X have norm 1, then we have

$$\frac{\partial R}{\partial \beta_j}(\beta) = -2X_j'(Y - X\beta) + \lambda\frac{\beta_j}{|\beta_j|}, \quad \forall \beta_j \neq 0.$$

Hence, we can see (after some easy computations) that $\beta_j \mapsto R(\beta_1, \ldots, \beta_{j-1}, \beta_j, \ldots, \beta_p)$ is minimum in

$$\beta_j = R_j\left(1 - \frac{\lambda}{2|R_j|}\right)_+$$

with $R_j = X_j'(Y - \sum_{k\neq j}\beta_k X_k)$.

The coordinate descent algorithm is summarized as follows:

- Initialise β_{init} arbitrarily
- Iterate until convergence:

$$\forall j = 1, \ldots, p, \beta_j = R_j \left(1 - \frac{\lambda}{2|R_j|}\right)_+$$

  with $R_j = X_j'(Y - \sum_{k \neq j} \beta_k X_k)$.
- Output β.

This algorithm is implemented in the R package `glmnet`.

Due to its parsimonious solution, this method is widely used to select variables in high dimension settings (when $p > n$).

5 Elastic Net

Elastic Net is a method that combines Ridge and Lasso regression, by introducing simultaneously the l_1 and l_2 penalties. The criterion to minimize is

$$\sum_{i=1}^{n}(Y_i - \beta_0 - \beta_1 X_i^{(1)} - \beta_2 X_i^{(2)} - \ldots - \beta_p X_i^{(p)})^2$$
$$+\lambda \left(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2 \right)$$

- For $\alpha = 1$, we recover the LASSO.
- For $\alpha = 0$, we recover the Ridge regression.

In this case, we have two tuning parameters to calibrate by cross-validation.

6 Principal Components Regression and Partial Least Square regression

6.1 Principal Component Regression (PCR)

We denote by $Z^{(1)}, \ldots Z^{(p)}$ the principal components associated to the variables $X^{(1)}, \ldots X^{(p)}$:

- $Z^{(1)}$ is the linear combination of $X^{(1)}, \ldots, X^{(p)}$ of the form $\sum_{i=1}^{p} \alpha_j X^{(j)}$ with $\sum \alpha_j^2 = 1$ with maximal variance.
- $Z^{(m)}$ is the linear combination of $X^{(1)}, \ldots, X^{(p)}$ of the form $\sum_{i=1}^{p} \alpha_{j,m} X^{(j)}$ with $\sum \alpha_{j,m}^2 = 1$ with maximal variance and orthogonal to $Z^{(1)}, \ldots, Z^{(m-1)}$.

The Principal Component Regression (PCR) consists in considering a predictor of the form:

$$\hat{Y}^{PCR} = \sum_{m=1}^{M} \hat{\theta}_m Z^{(m)}$$

with

$$\hat{\theta}_m = \frac{\langle Z^{(m)}, Y \rangle}{\|Z^{(m)}\|^2}.$$

Comments:

- If $M = p$, we keep all the variables and we recover the ordinary least square (OLS) estimator.
- If one can obtain a good prediction with $M < p$, then we have reduced the number of variables, hence the dimension.

- Nevertheless, interpretation is not always easy: if the variables are interpretable, the principal components (that correspond to linear combination of the variables) are generally difficult to interpret.
- This method is quite similar to the Ridge regression, which shrinks the coefficients of the principal components. Here, we set to 0 the coefficients of the principal components of order greater than M.
- The first principal components are not necessarily well correlated with the variable to explain Y, this is the reason why the PLS regression has been introduced.

6.2 Partial Least Square (PLS) regression

The principle of this method is to make a regression on linear combinations of the variables X_i's, that are highly correlated with Y.

- We assume that Y has been centered, and that the variables $X^{(j)}$ are also centered and normalized (with norm 1).
- The first PLS component is defined by:

$$W^{(1)} = \sum_{j=1}^{p} \langle Y, X^{(j)} \rangle X^{(j)}.$$

- The prediction associated to this first component is:

$$\hat{Y}^1 = \frac{\langle Y, W^{(1)} \rangle}{\|W^{(1)}\|^2} W^{(1)}.$$

Note that if the matrix X is orthogonal, this estimator corresponds to the ordinary least square (OLS) estimator, and in this case, the following steps of the PLS regression are useless.

- In order to obtain the following directions, we orthogonalize the variables $X^{(j)}$ with respect to the first PLS component $W^{(1)}$:
- We substract to each variables $X^{(j)}$ ($1 \leq j \leq p$) its orthogonal projection in the direction given by $W^{(1)}$ and we normalize the variables thus obtained.
- We compute the second PLS component $W^{(2)}$ in the same way as the first component by replacing the variables $X^{(j)}$'s by the new variables.
- We iterate this process by orthogonalizing at each step the variables with respect to the PLS components.

The algorithm is the following:

- $\hat{Y}^0 = \bar{Y}$ and $X^{(j),0} = X^{(j)}$. For $m = 1, \ldots, p$
- $W^{(m)} = \sum_{j=1}^{p} \langle Y, X^{(j,m-1)} \rangle X^{(j,m-1)}$.
- $\hat{Y}^m = \hat{Y}^{m-1} + \frac{\langle Y, W^{(m)} \rangle}{\|W^{(m)}\|^2} W^{(m)}$.
- $\forall j = 1, \ldots, p$, $X^{(j),m} = \frac{X^{(j),m-1} - \Pi_{W^{(m)}}(X^{(j),m-1})}{\|X^{(j),m-1} - \Pi_{W^{(m)}}(X^{(j),m-1})\|}$.
- The predictor $\hat{Y}^p$ obtained at step p corresponds to ordinary least square estimator.
- This method is useless if the variables $X^{(j)}$ are orthogonal.
- When the variables $X^{(j)}$ are correlated, PCR and PLS methods present the advantage to deal with new variables, that are orthogonal.
- The choice of the number of PCR or PLS components can be done by cross-validation.

- In general, the PLS method leads to more parcimoneous representations than the PCR method.
- The PLS regression leads to a **reduction of the dimension**.
- If p is large, this is particularly interesting, but can lead to problems of interpretation since the PLS components are linear combinations of the variables.
- There exists a sparse version: **sparse PLS** (inspired from the Lasso method), for which we consider linear combinations of the initial variables $X^{(j)}$ with only a few non zero coefficients, hence keeping only a few variables, which makes the interpretation more easy.

Chapter 5

Linear methods for classification, Linear Support Vector Machine

1 Introduction

In this chapter, we consider supervised classification problems. We have a data set with n observation points (or objects) $\boldsymbol{X}_i$ and their class (or label) Y_i. For example, the MNIST data set is a database of handwritten digits, where the objects $\boldsymbol{X}_i$ are images and $Y_i \in \{0, 1, \ldots, 9\}$. Many other examples can be considered, such as the recognition of an object in an image, the detection of spams for emails, the presence of some illness for patients (the observation points may be gene expression data) ... We have already seen in Chapter 2 the notion of best classifier, which is also called the Bayes classifier. A first generalized linear model, namely the **logistic regression** has been presented in Chapter 3. We propose here to study new linear methods for classification, first the **linear discriminant analysis** and the core of the chapter will be devoted to the linear **Support Vector Machine (SVM)**, which will be generalized to nonlinear SVM in Chapter 6.

2 Linear discriminant analysis

Let $(\boldsymbol{X}, Y)$ with unknown distribution P on $\mathcal{X} \times \mathcal{Y}$, where we assume that $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \{1, 2, \ldots, K\}$. We define

$$f_k(\boldsymbol{x}) = \mathbb{P}(Y = k / \boldsymbol{X} = \boldsymbol{x}).$$

A Bayes rule is defined by

$$f^*(\boldsymbol{x}) = \underset{k \in \{1,2,\ldots,K\}}{\operatorname{argmax}} f_k(\boldsymbol{x}).$$

We assume that the distribution of $\boldsymbol{X}$ has a density $f_{\boldsymbol{X}}$ and the distribution of $\boldsymbol{X}$ given $Y = k$ has a density g_k with respect to the Lebesgue measure on $\mathbb{R}^p$, and we set $\pi_k = \mathbb{P}(Y = k)$.

Exercise. — Prove that

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \sum_{l=1}^{K} g_l(\boldsymbol{x})\pi_l$$

and that

$$f_k(\boldsymbol{x}) = \frac{g_k(\boldsymbol{x})\pi_k}{\sum_{l=1}^{K} g_l(\boldsymbol{x})\pi_l}.$$

If we assume that the distribution of $\boldsymbol{X}$ given $Y = k$ is a multivariate normal distribution, with mean μ_k and covariance matrix Σ_k, we have

$$g_k(\boldsymbol{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\mu_k)'\Sigma_k^{-1}(\boldsymbol{x}-\mu_k)\right).$$

For the linear discriminant analysis, we furthermore assume that $\Sigma_k = \Sigma$ for all k. In this case we have

$$\log \mathbb{P}(Y = k/\boldsymbol{X} = \boldsymbol{x}) = C(\boldsymbol{x}) + \delta_k(\boldsymbol{x})$$

where $C(\boldsymbol{x})$ does not depend on the class k, and

$$\delta_k(\boldsymbol{x}) = \boldsymbol{x}'\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k'\Sigma^{-1}\mu_k + \log(\pi_k).$$

The Bayes rule will assign $\boldsymbol{x}$ to the class $f^*(\boldsymbol{x})$ which maximises $\delta_k(\boldsymbol{x})$.

$$\begin{aligned}\log\left(\frac{\mathbb{P}(Y=k/\boldsymbol{X}=\boldsymbol{x})}{\mathbb{P}(Y=l/\boldsymbol{X}=\boldsymbol{x})}\right) &= \log\left(\frac{\pi_k}{\pi_l}\right) + \boldsymbol{x}'\Sigma^{-1}(\mu_k-\mu_l) \\ &- \frac{1}{2}(\mu_k+\mu_l)'\Sigma^{-1}(\mu_k-\mu_l).\end{aligned}$$

Hence the decision boundary between the class k and the class l, $\{\boldsymbol{x}, \mathbb{P}(Y = k/\boldsymbol{X} = \boldsymbol{x}) = \mathbb{P}(Y = l/\boldsymbol{X} = \boldsymbol{x})\}$ is linear.

We want now to built a decision rule from a training sample $\boldsymbol{D}^n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ which is close to the Bayes rule. For this purpose, we have to estimate for all k π_k, μ_k and the matrix Σ. We consider the following estimators.

$$\hat{\pi}_k = \frac{N_k}{n},\ \hat{\mu}_k = \frac{\sum_{i=1}^{n} \boldsymbol{X}_i \mathbf{1}_{Y_i=k}}{N_k}$$

where $N_k = \sum_{i=1}^{n} \mathbf{1}_{Y_i=k}$. We estimate Σ by

$$\hat{\Sigma} = \sum_{k=1}^{K}\sum_{i=1}^{n} \frac{(\boldsymbol{X}_i - \hat{\mu}_k)(\boldsymbol{X}_i - \hat{\mu}_k)'\mathbf{1}_{Y_i=k}}{n-K}.$$

To conclude, the Linear Discriminant Analysis assigns the input $\boldsymbol{x}$ to the class $\hat{f}(\boldsymbol{x})$ which maximises $\hat{\delta}_k(\boldsymbol{x})$, where we have replaced in the expression of $\delta_k(\boldsymbol{x})$ the unknown quantities by their estimators.

Remark: If we no more assume that the matrix Σ does not depend on the class k, we obtain quadratic discriminant functions

$$\delta_k(x) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\boldsymbol{x}-\mu_k)'\Sigma_k^{-1}(\boldsymbol{x}-\mu_k) + \log(\pi_k).$$

This leads to the quadratic discriminant analysis.

3 Linear Support Vector Machine

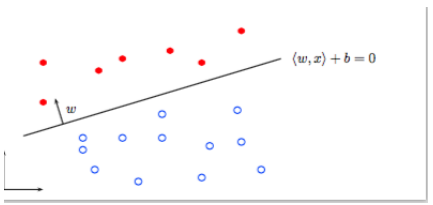
3.1 Linearly separable training set

We assume that $\mathcal{X} = \mathbb{R}^d$, endowed with the usual scalar product $\langle ., . \rangle$, and that $\mathcal{Y} = \{-1, 1\}$.

DEFINITION 12. — *The training set* $d_1^n = (x_1, y_1), \ldots, (x_n, y_n)$ *is called* **linearly separable** *if there exists* (w, b) *such that for all* i,
$y_i = 1$ *if* $\langle w, x_i \rangle + b > 0$,
$y_i = -1$ *if* $\langle w, x_i \rangle + b < 0$,
which means that

$$\forall i \; y_i \left(\langle w, x_i \rangle + b \right) > 0.$$

The equation $\langle w, x \rangle + b = 0$ defines a separating hyperplane with orthogonal vector w.

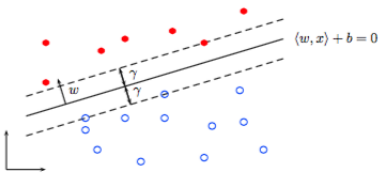


The function $f_{w,b}(x) = \mathbf{1}_{\langle w,x \rangle + b \geq 0} - \mathbf{1}_{\langle w,x \rangle + b < 0}$ defines a possible linear classification rule.

The problem is that there exists an infinity of separating hyperplanes, and therefore an infinity of classification rules.

Which one should we choose ? The response is given by Vapnik [40]. The classification rule with the best generalization properties corresponds to the separating hyperplane maximizing the margin γ between the two classes on the training set.

If we consider two entries of the training set, that are on the broder defining the margin, and that we call x_1 and x_{-1} with respective outputs 1 and -1, the separating hyperplane is located at the half-distance between x_1 and x_{-1}.



The margin is therefore equal to the half of the distance between x_1 and x_{-1} projected onto the normal vector of the separating hyperplane:

$$\gamma = \frac{1}{2} \frac{|\langle w, x_1 - x_{-1} \rangle|}{\|w\|}.$$

Let us notice that for all $\kappa \neq 0$, the couples $(\kappa w, \kappa b)$ and (w, b) define the same hyperplane.

DEFINITION 13. — *The hyperplane* $\langle w, x \rangle + b = 0$ *is* **canonical** *with respect to the set of vectors* $x_1, \ldots, x_k$ *if*

$$min_{i=1 \ldots k} |\langle w, x_i \rangle + b| = 1.$$

The separating hyperplane has the canonical form relatively to the vectors $\{x_1, x_{-1}\}$ *if it is defined by* (w, b) *where* $\langle w, x_1 \rangle + b = 1$ *and* $\langle w, x_{-1} \rangle + b = -1$. *In this case, we have* $\langle w, x_1 - x_{-1} \rangle = 2$, *hence*

$$\gamma = \frac{1}{\|w\|}.$$

3.2 A convex optimisation problem

Finding the separating hyperplane with maximal margin consists in finding (w, b) such that

$$\begin{array}{c} \|w\|^2 \text{ or } \frac{1}{2}\|w\|^2 \text{ is minimal} \\ \text{under the constraint} \\ y_i\left(\langle w, x_i\rangle + b\right) \geq 1 \text{ for all } i. \end{array}$$

This leads to a convex optimization problem with linear constraints, hence there exists a unique global minimizer.
The primal problem to solve is:

$$\text{Minimizing } \frac{1}{2}\|w\|^2 \text{ s. t. } y_i\left(\langle w, x_i\rangle + b\right) \geq 1 \; \forall\, i.$$

The corresponding **Lagrangian** is

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i \left(y_i\left(\langle w, x_i\rangle + b\right) - 1\right).$$

$$\begin{aligned} \frac{\partial L}{\partial w}(w, b, \alpha) &= w - \sum_{i=1}^{n} \alpha_i y_i x_i = 0 \Leftrightarrow w = \sum_{i=1}^{n} \alpha_i y_i x_i \\ \frac{\partial L}{\partial b}(w, b, \alpha) &= -\sum_{i=1}^{n} \alpha_i y_i = 0 \Leftrightarrow \sum_{i=1}^{n} \alpha_i y_i = 0 \end{aligned}$$

This leads to the **dual function**

$$\begin{aligned} \theta(\alpha) &= \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle + \sum_{i=1}^{n}\alpha_i - \sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle \\ &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle. \end{aligned}$$

The corresponding **dual problem** corresponds to the maximization of

$$\theta(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} \alpha_i\alpha_j y_i y_j \langle x_i, x_j\rangle$$

under the constraint $\sum_{i=1}^{n} \alpha_i y_i = 0$ and $\alpha_i \geq 0\ \forall i$.

The **Karush-Kuhn-Tucker conditions** are

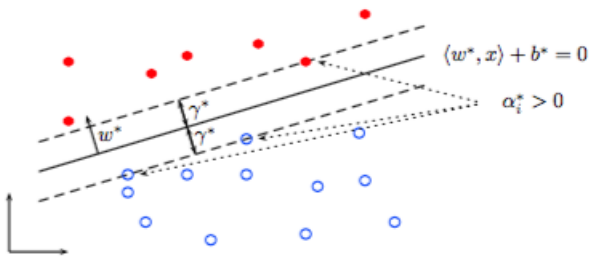
- $\alpha_i^* \geq 0\ \forall i = 1 \ldots n.$
- $y_i\left(\langle w^*, x_i\rangle + b^*\right) \geq 1\ \forall i = 1 \ldots n.$
- $\alpha_i^*\left(y_i\left(\langle w^*, x_i\rangle + b^*\right) - 1\right) = 0\ \forall\, i = 1 \ldots n.$ (complementary condition)

The solution α^* of the dual problem can be obtained with classical optimization softwares.
Remarks : The only pertinent information from the observations $(x_i)_{1\leq i\leq n}$ to solve the problem is the Gram matrix $G = (\langle x_i, x_j\rangle)_{1\leq i,j\leq n}$.
The solution does not depend on the dimension d, but depends on the sample size n, hence it is interesting to notice that when $\mathcal{X}$ is high dimensional, linear SVM do not suffer from the curse of dimensionality.

3.3 Supports Vectors

Only the $\alpha_i^* > 0$ are involved in the definition of $w^* = \sum_{i=1}^{n} \alpha_i y_i x_i$. If the number of values $\alpha_i^* > 0$ is small, the solution of the dual problem is called **"sparse"**.

DEFINITION 14. — *The x_i such that $\alpha_i^* > 0$ are called the* **support vectors**. *They are located on the border defining the maximal margin namely $y_i\left(\langle w^*, x_i\rangle + b^*\right) = 1$ (c.f. complementary KKT condition).*



We finally obtain the following classification rule:

$$\hat{f}(x) = \mathbf{1}_{\langle w^*, x\rangle + b^* \geq 0} - \mathbf{1}_{\langle w^*, x\rangle + b^* < 0},$$

with

- $w^* = \sum_{i=1}^n \alpha_i^* x_i y_i$,
- $b^* = -\frac{1}{2}\{\min_{y_i=1}\langle w^*, x_i\rangle + \min_{y_i=-1}\langle w^*, x_i\rangle\}$.

The maximal margin equals $\gamma^* = \frac{1}{\|w^*\|} = \left(\sum_{i=1}^n (\alpha_i^*)^2\right)^{-1/2}$ (provided the x_i's are normalized).

The α_i^* that do not correspond to support vectors (sv) are equal to 0, and therefore

$$\hat{f}(x) = \mathbf{1}_{\sum_{x_i sv} y_i \alpha_i^* \langle x_i, x\rangle + b^* \geq 0} - \mathbf{1}_{\sum_{x_i sv} y_i \alpha_i^* \langle x_i, x\rangle + b^* < 0}.$$

The previous formulation has two main drawbacks : it assumes that the classes are linearly separable and it is also very sensitive to outliers as illustrated in Figure 5.1.

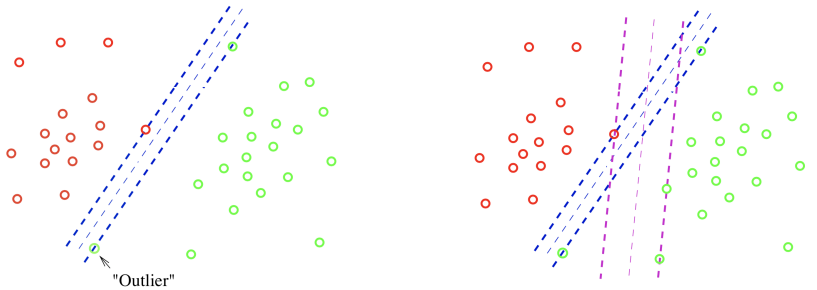


Figure 5.1: Lack of robustness of SVM's in the separable case

3.4 Flexible margin

In the general case, we allow some points to be in the margin and even on the wrong side of the margin. We introduce the slack variable $\xi = (\xi_1, \ldots, \xi_n)$ and the constraint $y_i(\langle w, x_i\rangle + b) \geq 1$ becomes $y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i$, with $\xi_i \geq 0$.

- If $\xi_i \in [0, 1]$ the point is well classified but in the region defined by the margin.
- If $\xi_i > 1$ the point is misclassified.

The margin is called **flexible margin**.

3.5 Optimization problem with relaxed constraints

In order to avoid too large margins, we penalize large values for the slack variable ξ_i.

The **primal optimization problem** is formalized as follows :

$$\text{Minimize with respect to } (w, b, \xi) \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i \text{ such that}$$
$$y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \; \forall \, i$$
$$\xi_i \geq 0$$

Remarks :

- $C > 0$ is a tuning parameter of the SVM algorithm. It will determine the tolerance to misclassifications. If C increases, the number of misclassified points decreases, and if C decreases, the number of misclassified points increases. C is generally calibrated by cross-validation.

Exercise. — Write the Lagrangian, the dual problem, and the KKT conditions.

Karush-Kuhn-Tucker conditions :

- $0 \leq \alpha_i^* \leq C \; \forall i = 1 \ldots n.$
- $y_i\left(\langle w^*, x_i\rangle + b^*\right) \geq 1 - \xi_i^* \; \forall i = 1 \ldots n.$
- $\alpha_i^*\left(y_i\left(\langle w^*, x_i\rangle + b^*\right) + \xi_i^* - 1\right) = 0 \; \forall \, i = 1 \ldots n.$
- $\xi_i^*(\alpha_i^* - C) = 0.$

As previously, we obtain the following classification rule:

$$\hat{f}(x) = \mathbf{1}_{\langle w^*, x\rangle + b^* \geq 0} - \mathbf{1}_{\langle w^*, x\rangle + b^* < 0},$$

with

- $w^* = \sum_{i=1}^n \alpha_i^* x_i y_i,$
- $b^* = -\frac{1}{2}\left\{\min_{y_i=1}\langle w^*, x_i\rangle + \min_{y_i=-1}\langle w^*, x_i\rangle\right\}.$

We have here two types of support vectors (x_i such that $\alpha_i^* > 0$) :

- The support vectors for which the slack variables are equal to 0. They are located on the border of the region defining the margin.
- The support vectors for which the slack variables are not equal to 0: $\xi_i^* > 0$ and in this case $\alpha_i^* = C$.

For the vectors that are not support vectors, we have $\alpha_i^* = 0$ and $\xi_i^* = 0$.

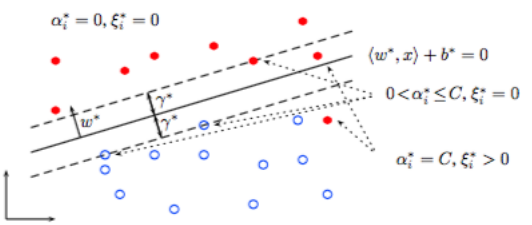


Figure 5.2: Support Vectors in the non separable case

We have assumed in this chapter that the classes are (nearly) linearly separable. This assumption is often unrealistic, and we will see in the Chapter 6 how to extend the SVM classifiers to a more general setting. Moreover, we focused here on classification problems but procedures based on support vector for regression have also been proposed and will be presented in Chapter 6.

Chapter 6

Kernel methods: Support Vector Machines and Support Vector Regression

1 Introduction

In Chapter 5, we have studied linear SVM. The assumption was that the training set is nearly linearly separable. In most cases, this assumption is not realistic.

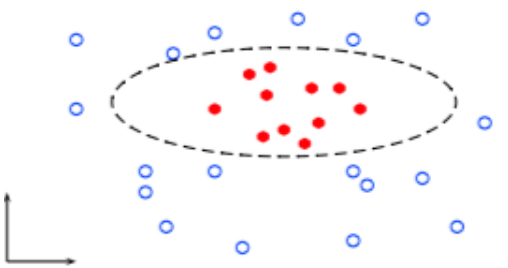


Figure 6.1: Non linearly separable training set

In this case, a linear SVM leads to bad performances and a high number of support vectors. We can make the classification procedure more flexible by enlarging the feature space and sending the entries $\{x_i, i = 1 \dots n\}$ in an Hilbert space $\mathcal{H}$, with high or possibly infinite dimension, via a function ϕ, and we apply a linear SVM procedure on the new training set $\{(\phi(x_i), y_i), i = 1 \dots n\}$. The space $\mathcal{H}$ is called the **feature space**. This idea is due to Boser, Guyon, Vapnik (1992).
In Figure 6.1, setting $\phi(x) = (x_1^2, x_2^2, x_1, x_2)$, the training set becomes linearly separable in $\mathbb{R}^4$, and a linear SVM is appropriate.

2 The kernel trick

A natural question arises: how can we choose $\mathcal{H}$ and ϕ ? In fact, we do not choose $\mathcal{H}$ and ϕ but a *kernel* .
The classification rule is

$$\hat{f}(x) = \mathbf{1}_{\sum y_i \alpha_i^* (\langle \phi(x_i), \phi(x) \rangle + b^*) \geq 0} - \mathbf{1}_{\sum y_i \alpha_i^* (\langle \phi(x_i), \phi(x) \rangle + b^*) < 0},$$

where the α_i^*'s are the solutions of the dual problem in the feature space $\mathcal{H}$:

Maximizing $\theta(\alpha) = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j \langle \phi(x_i), \phi(x_j) \rangle$
s. t. $\sum_{i=1}^n \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C \; \forall i$.

It is important to notice that the final classification rule in the feature space depends on ϕ only through scalar products of the form $\langle \phi(x_i), \phi(x) \rangle$ or $\langle \phi(x_i), \phi(x_j) \rangle$.
The only knowledge of the function k defined by $k(x, x') = \langle \phi(x), \phi(x') \rangle$ allows to define the SVM in the feature space $\mathcal{H}$ and to derive a classification rule in the space $\mathcal{X}$. The explicite computation of ϕ is not required.

DEFINITION 15. — *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle$ for a given function $\phi : \mathcal{X} \to \mathcal{H}$ is called a* **kernel**.

A kernel is generally more easy to compute than the function ϕ that returns values in a high dimensional space. For example, for $x = (x_1, x_2) \in \mathbb{R}^2$, $\phi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$, and $k(x, x') = \langle x, x' \rangle^2$.
Let us now give a property to ensure that a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ defines a kernel.

PROPOSITION 7. — **Mercer condition** *If the function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is continuous, symmetric, and if for all finite subset $\{x_1, \ldots, x_k\}$ in $\mathcal{X}$, the matrix $(k(x_i, x_j))_{1 \leq i,j \leq k}$ is positive definite:*

$$\forall c_1, \ldots, c_k \in \mathbb{R}, \sum_{i,j=1}^k c_i c_j k(x_i, x_j) \geq 0,$$

then, there exists an Hilbert space $\mathcal{H}$ and a function $\phi : \mathcal{X} \to \mathcal{H}$ such that $k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$. The space $\mathcal{H}$ is called the **Reproducing Kernel Hilbert Space (RKHS)** *associated to k.*
We have:

1. *For all $x \in \mathcal{X}$, $k(x, .) \in \mathcal{H}$ where $k(x, .) : y \mapsto k(x, y)$.*
2. **Reproducing property***:*

$$h(x) = \langle h, k(x, .) \rangle_{\mathcal{H}} \text{ for all } x \in \mathcal{X} \text{ and } h \in \mathcal{H}.$$

Let us give some examples. The Mercer condition is often hard to verify but we know some classical examples of kernels that can be used. We assume that $\mathcal{X} = \mathbb{R}^d$.

- **Linear kernel** : $k(x, x') = \langle x, x' \rangle$.
- p **degree polynomial kernel** : $k(x, x') = (1 + \langle x, x' \rangle)^p$.
- **Gaussian kernel (RBF)**: $k(x, x') = e^{-\frac{\|x - x'\|^2}{2\sigma^2}}$. ϕ returns values in a infinite dimensional space.
- **Laplacian kernel** : $k(x, x') = e^{-\frac{\|x - x'\|}{\sigma}}$.
- **Sigmoid kernel** : $k(x, x') = \tanh(\kappa \langle x, x' \rangle + \theta)$ (this kernel is not positive definite).

By way of example, let us precise the RKHS associated with the Gaussian kernel.

PROPOSITION 8. — *For any function $f \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$ and $\omega \in \mathbb{R}^d$, we define the Fourier transform*

$$\boldsymbol{F}[f](\omega) = \frac{1}{(2\pi)^{d/2}} \int_{\mathbb{R}^d} f(t) e^{-i\langle \omega, t \rangle} dt.$$

For any $\sigma > 0$, the functional space

$$\mathcal{H}_\sigma = \{f \in C_0(\mathbb{R}^d) \cap L^1(\mathbb{R}^d) \text{ such that } \int_{\mathbb{R}^d} |\boldsymbol{F}[f](\omega)|^2 e^{\sigma^2 |\omega|^2/2} d\omega < +\infty\}$$

endowed with the scalar product

$$\langle f, g \rangle_{\mathcal{H}_\sigma} = (2\pi\sigma^2)^{-d/2} \int_{\mathbb{R}^d} \overline{\boldsymbol{F}[f](\omega)} \boldsymbol{F}[g](\omega) e^{\sigma^2 |\omega|^2/2} d\omega,$$

is the RKHS associated with the Gaussian kernel $k(x, x') = e^{-\frac{\|x-x'\|^2}{2\sigma^2}}$.

Indeed, for all $x \in \mathbb{R}^d$, the function $k(x, .)$ belongs to $\mathcal{H}_\sigma$ and we have

$$\langle h, k(x, .) \rangle_{\mathcal{H}_\sigma} = \boldsymbol{F}^{-1}[\boldsymbol{F}[h]](x) = h(x).$$

The RKHS $\mathcal{H}_\sigma$ contains very regular functions, and the norm $\|h\|_{\mathcal{H}_\sigma}$ controls the smoothness of the function h. When σ increases, the functions of the RKHS become smoother. See A. Smola and B. Scholkopf [34] for more details on RKHS.

We have seen some examples of kernels. One can construct new kernels by aggregating several kernels. For example let k_1 and k_2 be two kernels and f a function $\mathbb{R}^d \to \mathbb{R}$, $\phi : \mathbb{R}^d \to \mathbb{R}^{d'}$, B a positive definite matrix, P a polynomial with positive coefficients and $\lambda > 0$.

The functions defined by $k(x, x') = k_1(x, x') + k_2(x, x')$, $\lambda k_1(x, x')$, $k_1(x, x')k_2(x, x')$, $f(x)f(x')$, $k_1(\phi(x), \phi(x'))$, $x^T B x'$, $P(k_1(x, x'))$, or $e^{k_1(x,x')}$ are still kernels.

We have presented examples of kernels for the case where $\mathcal{X} = \mathbb{R}^d$ but a very interesting property is that kernels can be defined for very general input spaces, such as sets, trees, graphs, texts, DNA sequences ...

3 Minimization of the convexified empirical risk

The ideal classification rule is the one which minimizes the risk $L(f) = \mathbb{P}(Y \neq f(X))$, we have seen that the solution is the Bayes rule f^*. A classical way in nonparametric estimation or classification problems is to replace the risk by the empirical risk and to minimize the empirical risk:

$$L_n(f) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{Y_i \neq f(X_i)}.$$

In order to avoid overfitting, the minimization is restricted to a set $\mathcal{F}$:

$$\hat{f} = \text{argmin}_{f \in \mathcal{F}} L_n(f).$$

The risk of $\hat{f}$ can be decomposed in two terms:

$$0 \leq L(\hat{f}) - L(f^*) = \min_{f \in \mathcal{F}} L(f) - L(f^*) + L(\hat{f}) - \min_{f \in \mathcal{F}} L(f).$$

The first term $\min_{f \in \mathcal{F}} L(f) - L(f^*)$ is the approximation error, or bias term, the second term $L(\hat{f}) - \min_{f \in \mathcal{F}} L(f)$ is the stochastic error or variance

term. Enlarging the class $\mathcal{F}$ reduces the approximation error but increases the stochastic error.
The empirical risk minimization classifier cannot be used in practice because of its computational cost, indeed L_n is not convex. This is the reason why we generally replace the empirical misclassification probability L_n by some convex surrogate, and we consider convex classes $\mathcal{F}$. We consider a loss function ℓ, and we require the condition $\ell(z) \geq \mathbf{1}_{z<0}$, which will allow to give an upper bound for the misclassification probability; indeed

$$\mathbb{E}(\ell(Yf(X))) \geq \mathbb{E}(\mathbf{1}_{Yf(X)<0}) = \mathbb{P}(Y \neq f(X)).$$

Classical convex losses ℓ are the hinge loss $\ell(z) = (1-z)_+$, the exponential loss $\ell(z) = \exp(-z)$, the logit loss $\ell(z) = \log_2(1+\exp(-z))$.

Let us show that SVM are solutions of the minimization of the convexified (with the hinge loss) and penalized empirical risk. For the sake of simplicity, we consider the linear case.

We first notice that the following optimization problem: Minimizing $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \xi_i$ s. t. $\begin{cases} y_i\left(\langle w, x_i\rangle + b\right) \geq 1-\xi_i \ \forall\, i \\ \xi_i \geq 0 \end{cases}$

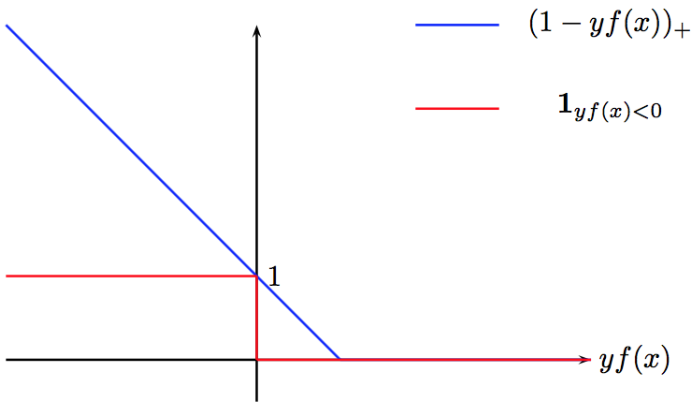
is equivalent to minimize

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^n \left(1 - y_i\left(\langle w, x_i\rangle + b\right)\right)_+,$$

or equivalently

$$\frac{1}{n}\sum_i^n \left(1 - y_i\left(\langle w, x_i\rangle + b\right)\right)_+ + \frac{1}{2Cn}\|w\|^2.$$

$\gamma(w, b, x_i, y_i) = \left(1 - y_i\left(\langle w, x_i\rangle + b\right)\right)_+$ is a convex upper bound of the empirical risk $\mathbf{1}_{y_i(\langle w, x_i\rangle + b)<0}$ with the hinge loss.



Hence, SVM are solutions of the minimization of the convexified empirical risk with the hinge loss ℓ plus a penalty term. Indeed, SVM are solutions of

$$\text{argmin}_{f\in\mathcal{F}} \frac{1}{n}\sum_{i=1}^n \ell(y_i f(x_i)) + \text{pen}(f),$$

where

$$\mathcal{F} = \{\langle w, x\rangle + b, w \in \mathbb{R}^d, b \in \mathbb{R}\}$$

and

$$\forall f \in \mathcal{F}, \text{pen}(f) = \frac{1}{2Cn}\|w\|^2.$$

4 Support Vector Regression

Although the framework of the chapter is classification, let us mention that kernel methods can also be used for regression function estimation.

Suppose we have a training sample $\{(x_1, y_1), \ldots, (x_n, y_n)\} \in (\mathcal{X} \times \mathbb{R})^n$, where $\mathcal{X}$ denotes the space of the inputs (for example $\mathcal{X} = \mathbb{R}^d$). The ε support vector regression, introduced by Vapnik (1995) aims to find a function f such that for all i, the deviation between $f(x_i)$ and y_i is at most ε, and such that, at the same time, f is as flat as possible. Let us first consider the case of linear predictors :

$$f(x) = \langle w, x \rangle + b, \text{ with } x \in \mathcal{X}, b \in \mathbb{R}.$$

Flatness means here that $\|w\|$ is small. This leads to the convex optimization problem :

$$\begin{array}{c} \text{Minimize } \frac{1}{2}\|w\|^2 \\ \text{under the constraints, for all } i \end{array}$$

$$\left\{ \begin{array}{c} y_i - (\langle w, x_i \rangle + b) \leq \varepsilon \\ -y_i + (\langle w, x_i \rangle + b) \leq \varepsilon \end{array} \right.$$

Note that here, we do not care of errors less than ε, but we do not accept errors greater than ε. The tacit assumption is that the above problem admits a solution. To be more general, we want to allow some errors. Like for the classification problem, we introduce slack variables ξ_i, ξ_i' to overcome possible unfeasible constraints in the previous optimization problem. This leads to the following formulation :

$$\begin{array}{c} \text{Minimize } \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i') \\ \text{under the constraints, for all } i \end{array} \tag{6.1}$$

$$\left\{ \begin{array}{c} y_i - (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i \\ -y_i + (\langle w, x_i \rangle + b) \leq \varepsilon + \xi_i' \\ \xi_i, \xi_i' \geq 0 \end{array} \right.$$

Exercise. — Prove that this optimization problem is equivalent to the minimization of

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\ell_\varepsilon(y_i, f(x_i)),$$

where ℓ_ε is the so-called ε-insensitive loss function defined by

$$\begin{array}{rcll} \ell_\varepsilon(y, y') & = & 0 & \text{if } |y - y'| \leq \varepsilon \\ & = & |y - y'| - \varepsilon & \text{otherwise .} \end{array}$$

Draw a picture to represent the support vector regression problem in the linear case.

In most cases, the optimization problem 6.1 can be solved more easily in its dual formulation. Moreover, like for classification problems, the dual formulation allows to extend easily the support vector regression to nonlinear functions. The Lagrangian is

$$L(w, b, \xi, \xi', \eta, \eta', \alpha, \alpha') = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i') - \sum_{i=1}^{n}(\eta_i\xi_i + \eta_i'\xi_i')$$
$$- \sum_{i=1}^{n}\alpha_i(\varepsilon + \xi_i - y_i + \langle w, x_i \rangle + b) - \sum_{i=1}^{n}\alpha_i'(\varepsilon + \xi_i' + y_i - \langle w, x_i \rangle - b),$$

with $\eta, \eta', \alpha, \alpha'$ the Lagrange multipliers, $\eta_i, \eta_i', \alpha_i, \alpha_i' \geq 0$.

The cancellation of the partial derivatives with respect to the primal variables $\frac{\partial L}{\partial w}(w, b, \xi, \xi', \eta, \eta', \alpha, \alpha')$, $\frac{\partial L}{\partial b}(w, b, \xi, \xi', \eta, \eta', \alpha, \alpha')$ and $\frac{\partial L}{\partial \xi_i^{(\prime)}}(w, b, \xi, \xi', \eta, \eta', \alpha, \alpha')$ leads to the following dual problem.

Dual problem. Show that the dual problem can be formulated as follows :

$$\begin{aligned} \text{Maximize} \quad & -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i-\alpha_i')(\alpha_j-\alpha_j')\langle x_i,x_j\rangle \\ & -\varepsilon\sum_{i=1}^{n}(\alpha_i+\alpha_i')+\sum_{i=1}^{n}y_i(\alpha_i-\alpha_i') \\ \text{subject to} \quad & \sum_{i=1}^{n}(\alpha_i-\alpha_i')=0 \text{ and } 0\le\alpha_i,\alpha_i'\le C\ \forall i. \end{aligned}$$

Karush-Kuhn-Tucker conditions :

- $\alpha_i^*(\varepsilon+\xi_i^*-y_i+\langle w^*,x_i\rangle+b^*)=0$
- $(\alpha_i')^*(\varepsilon+(\xi_i')^*+y_i-\langle w^*,x_i\rangle-b^*)=0$
- $\xi_i^*(C-\alpha_i^*)=0,\ (\xi_i')^*(C-(\alpha_i')^*)=0$

Exercise. — Draw a picture similar to Figure 5.2 to show the support vectors for the regression problem.

As previously, only the scalar product $\langle x_i,x_j\rangle$ are involved in the solution, allowing easily to extend to nonlinear regression functions.

5 Kernel Regression Least Square

We present here another regression method based on kernels : the Kernel Regression Least Square procedure. It is based on a penalized least square criterion. Let $(\boldsymbol{X_i},Y_i)_{1\le i\le n}$ the observations, with $\boldsymbol{X_i}\in\mathbb{R}^p$, $Y_i\in\mathbb{R}$. We consider a positive definite kernel k defined on $\mathbb{R}^p$:

$$k(\boldsymbol{x},\boldsymbol{y})=k(\boldsymbol{y},\boldsymbol{x});\quad \sum_{i,j=1}^{n}c_ic_jk(\boldsymbol{X_i},\boldsymbol{X_j})\ge 0.$$

We are looking for a predictor of the form

$$f(\boldsymbol{x})=\sum_{i=1}^{n}c_jk(\boldsymbol{X_j},\boldsymbol{x}),\ \boldsymbol{c}\in\mathbb{R}^n.$$

Let us denote by $\boldsymbol{K}$ the matrix defined by $\boldsymbol{K}_{i,j}=k(\boldsymbol{X_i},\boldsymbol{X_j})$. The KRLS method consists in minimizing for f on the form defined above the penalized least square criterion

$$\sum_{i=1}^{n}(Y_i-f(\boldsymbol{X_i}))^2+\lambda\|f\|_{\boldsymbol{K}}^2,$$

where

$$\|f\|_{\boldsymbol{K}}^2=\sum_{i,j=1}^{n}c_ic_jk(\boldsymbol{X_i},\boldsymbol{X_j}).$$

Equivalently, we minimize for $c\in\mathbb{R}^n$ the criterion

$$\|\boldsymbol{Y}-\boldsymbol{K}\boldsymbol{c}\|^2+\lambda\boldsymbol{c}'\boldsymbol{K}\boldsymbol{c}.$$

There exists an explicit solution

$$\hat{\boldsymbol{c}}=(\boldsymbol{K}+\lambda I_n)^{-1}Y,$$

which leads to the predictor

$$\hat{f}(\boldsymbol{x})=\sum_{j=1}^{n}\hat{c}_jk(\boldsymbol{X_j},\boldsymbol{x}).$$

$$\hat{\boldsymbol{Y}}=\boldsymbol{K}\hat{\boldsymbol{c}}.$$

With a kernel corresponding to the scalar product, we recover a linear predictor

$$\boldsymbol{K}=\boldsymbol{X}\boldsymbol{X}',\hat{\boldsymbol{c}}=(\boldsymbol{X}\boldsymbol{X}'+\lambda I_n)^{-1}Y,$$

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{n} \hat{c}_j \langle \boldsymbol{X_j}, \boldsymbol{x} \rangle.$$

For polynomial or Gaussian kernels for example, we obtain non linear predictors. As for SVM, an important interest of this method is the possibility to be generalized to complex predictors such as text, graphs, DNA sequences .. as soon as one can define a kernel function on such objects.

6 Conclusion

- Using kernels allows to delinearize classification algorithms by mapping $\mathcal{X}$ in the RKHS $\mathcal{H}$ with the map $x \mapsto k(x, .)$. It provides nonlinear algorithms with almost the same computational properties as linear ones.
- SVM have nice theoretical properties, cf. Vapnik's theory for empirical risk minimization [40].
- The use of RKHS allows to apply to any set $\mathcal{X}$ (such as set of graphs, texts, DNA sequences ..) algorithms that are defined for vectors as soon as we can define a kernel $k(x, y)$ corresponding to some measure of similarity between two objects of $\mathcal{X}$.
- Important issues concern the choice of the kernel, and of the tuning parameters to define the SVM procedure.
- Note that SVM can also be used for multi-class classification problems for example, one can built a SVM classifier for each class against the others and predict the class for a new point by a majority vote.
- Kernels methods are also used for non supervised classification (kernel PCA), and for anomaly detection (One-class SVM).

Chapter 7

Classification and Regression Trees

1 Introduction

The recursive partitioning or segmentation methods were first introduced in the 1960's. The method studied in this course was presented in a paper by Breiman et al [9] in 1984 under the acronym of CART for Classification and Regression Trees. As indicated by its name, this method can be used either for regression or for classification. The CART algorithm is a non parametric method to build estimators in a multidimensional framework. The method, based on trees, relies on a partition of the space of input variables. We then infer a simple model (constant piecewise functions in regression and a single class in classification) on each element of the partition. The obtained solutions can be represented in a graphic with a tree that is very easy to interpret. The trees are based on a recursive sequence of division rules or splits, each of them based on a single explanatory variable.

Figure 1 shows an illustrative example of a classification tree. The variables `Age`, `Income` and `Sex` are used partition the observations with tree structure. All the observations are gathered at the root of the tree then each division or cut

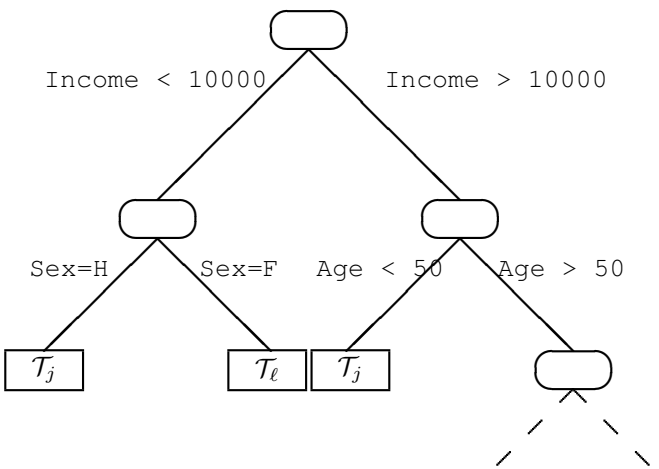


Figure 7.1: Elementary example of classification tree.

separates each node into two child nodes more *homogeneous* than the parent node in the sense of a *criterion* to be specified and depending on the type of the variable Y that we have: quantitative or qualitative.

A first very simple and natural non parametric procedure in supervised regression or classification is the k-Nearest Neighbors (k-NN) method. Given a leaning sample $\{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ in $\mathcal{X} \times \mathcal{Y}$, we want to predict the output Y associated to a new entry $\boldsymbol{x}$. For this, it seems natural to built the predictor from the observations in the training sample that are "close" to $\boldsymbol{x}$. We consider a distance d on $\mathcal{X}$. We fix an integer k and we retain the k nearest to $\boldsymbol{x}$ observations $\{\boldsymbol{X}_{(1)}, \ldots, \boldsymbol{X}_{(k)}\}$ and the associated outputs $(Y_{(1)}, \ldots, Y_{(k)})$. In a regression context, the prediction at point $\boldsymbol{x}$ is obtained from the mean of the observations $(Y_{(1)}, \ldots, Y_{(k)})$ while in classification we consider a majority vote. The choice of k is of course crucial. A too small value leads to overfitting (small bias but high variance) while a large value of k may lead to underfitting (small variance but probably high bias).

CART will use the same idea of local mean or majority vote, but the cell in $\mathcal{X}$ that is used to predict at point $\boldsymbol{x}$ is obtained from a more sophisticated way than simply considering the k-Nearest Neighbors of $\boldsymbol{x}$ in the learning sample. It will also take into account the values of the Y_i's. When partitioning ends, each terminal node of the complete tree becomes a leaf to which is assigned a value if Y is quantitative and a class if Y is qualitative.

The last step consists in pruning the complete tree, which corresponds to a model selection procedure in order to reduce the complexity and avoid overfitting. Since Breiman et al. (1984) [9] have introduced this algorithm, CART have been very successful with the major advantage of an easy interpretation of the trees. The drawback is that these models are particularly unstable (not robust), very sensitive to fluctuations in the training sample. Furthermore, for quantitative explanatory variables, the construction of a tree constitutes a *dyadic partitioning* of space (see Figure 7.2). The model thus defined is, by construction, discontinuous which may be a problem if the phenomenon to be modeled is regular.

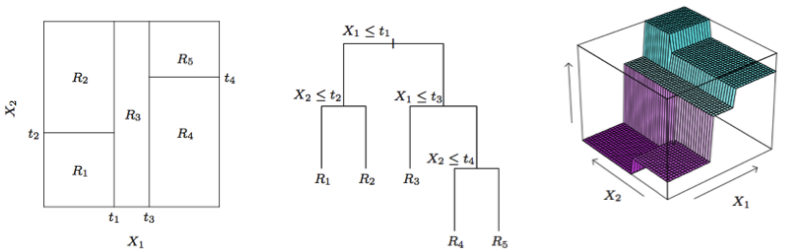


Figure 7.2: Source: Hastie, Tibshirani, Friedman (2019), "The elements of statistical learning"

These two aspects or weaknesses of CART: instability and irregularities are at the origin of the success of the methods of aggregation leading to Random Forests proposed by Breiman (2001) [8], that will be the topic of next chapter.

2 Construction of a maximal binary tree

We observe p quantitative or qualitative explanatory variables X^j and a variable to predict Y which is either qualitative with m modalities $\{\mathcal{T}_\ell; \ell = 1 \ldots, m\}$ or real quantitative, on a sample of n individuals.

The construction of a binary discrimination tree (cf. figure 1) consists in determining a sequence of *nodes*.

- A node is defined by the choice of a variable among the p explanatory variables and of a *division* which induces a partition into two classes.

Implicitly, to each node corresponds a subset of the initial sample to which a dichotomy is applied.

- A division is defined by a *threshold value* if the selected variable is quantitative or a split into two *groups of modalities* if the variable is qualitative.
- At the root, the initial node corresponds to the whole sample; the procedure is then iterated over each of the subsets.

The algorithm requires:

1. the definition of a *criterion* allowing to select the best division among all *admissible* ones for the different variables;
2. a rule allowing to decide that a node is terminal: it thus becomes a *leaf*;
3. the predicted value (class or real value) associated to a leaf.

2.1 Division criteria

A division is said to be *admissible* if the two corresponding son nodes are not empty. If the explanatory variable is a quantitative variable with m possible values (or qualitative but ordinal with m modalities), it provides $(m-1)$ possible binary divisions. If it qualitative but not ordinal, the number of divisions becomes $2^{(m-1)}-1$.

*W*arning : the algorithm tends to favor the selection of explanatory variables with many modalities because they offer more flexibility in the construction of two subgroups. These variables should be used carefully (e.g. the postal code) because they are likely to favor overfitting; it is often preferable to drastically reduce the number of modalities (e.g. geographic region or urban zone *vs.* rural zone) by merging modalities, which is classical in multiple correspondence analysis for example.

The division criterion is based on the definition of an *heterogeneity* function presented in the next section. The objective is to divide the observations which compose a node into two more homogeneous groups with respect to the variable to explain Y.

Dividing the node κ creates two son nodes. For simplicity, they are denoted κ_L (left node) and κ_R (right node).

Among all the admissible divisions of the node κ, the algorithm retains the one which minimizes the sum of the heterogeneities of the son nodes $D_{\kappa_L} + D_{\kappa_R}$. This amounts to solving at each node κ:

$$\max_{\{\textit{divisions of} X^j; j=1,p\}} D_\kappa - (D_{\kappa_L} + D_{\kappa_R})$$

Graphically, the length of each branch can be represented proportionally to the reduction in heterogeneity induced by the division.

2.2 Stopping rule

The growth of the tree stops at a given node, which therefore becomes a terminal node also called a *leaf*, when it is homogeneous (all the individuals have the same value for Y) or when there is no longer an admissible partition or (to avoid unnecessarily fine splittings) when the number of observations it contains is less than some prescribed value (generally chosen between 1 and 5).

2.3 Assignment

When Y is quantitative, the predicted value associated to a leaf is the average of the values of the Y_i's among the observations belonging to this terminal node. In the qualitative case, each leaf or terminal node is assigned to a class $\mathcal{T}_\ell$ of Y by a majority vote.

3 Homogeneity criterion

3.1 Constructing regression trees

For a given region (node) κ with cardinality $|\kappa|$, we define the empirical variance at node $|\kappa|$ by

$$V_\kappa = \frac{1}{|\kappa|}\sum_{i\in\kappa}(Y_i - \overline{Y}_\kappa)^2,$$

where $\overline{Y}_\kappa = \frac{1}{|\kappa|}\sum_{i\in\kappa} Y_i$.
The heterogeneity at the node κ is then defined by

$$D_\kappa = \sum_{i\in\kappa}(Y_i - \overline{Y}_\kappa)^2 = |\kappa| V_\kappa$$

Splitting procedure: For a variable x_j, and a split candidate t, define left and right subregions

$$\kappa_L(t,j) = \{X^j \leq t\}, \qquad \kappa_R(t,j) = \{X^j > t\}.$$

Find (j,t) in order to minimize

$$J(j,t) = D_{\kappa_L(t,j)} + D_{\kappa_R(t,j)},$$

or equivalently to maximize the decrease in heterogeneity

$$D_\kappa - J(j,t)$$

Figure 7.3 provides an illustration in dimension 1.

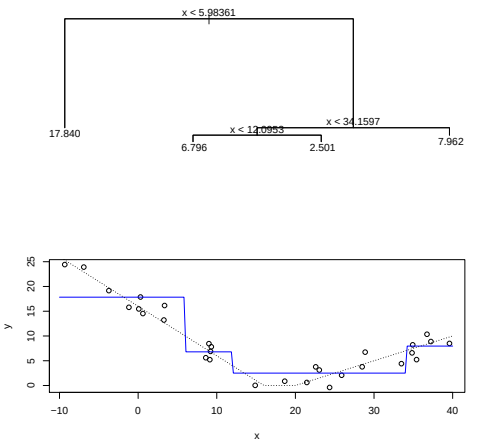


Figure 7.3: A regression tree in dimension 1

3.2 Constructing classification trees

We use the same procedure, with specific notions of **heterogeneity measures in classification**. Two main measures are considered to define the heterogeneity of node κ.
For $\ell = 1, \ldots, m$, let p_κ^ℓ denote proportion of the class $\mathcal{T}_\ell$ of Y in the node κ.

- The Cross-Entropy or deviance is defined by

$$E_\kappa = -\sum_{\ell=1}^{m} p_\kappa^\ell \log(p_\kappa^\ell).$$

  The heterogeneity at the node κ is then defined by

$$D_\kappa = -|\kappa| \sum_{\ell=1}^{m} p_\kappa^\ell \log(p_\kappa^\ell).$$

  The cross-entropy is maximal in $(\frac{1}{m}, \ldots, \frac{1}{m})$, minimal in $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$
  (by continuity, we assume that $0 \log(0) = 0$).

- The Gini concentration s defined by

$$G_\kappa = \sum_{\ell=1}^{m} p_\kappa^\ell (1 - p_\kappa^\ell),$$

  which leads to the heterogeneity at the node κ

$$D_\kappa = |\kappa| \sum_{\ell=1}^{m} p_\kappa^\ell (1 - p_\kappa^\ell).$$

An illustration of these two heterogeneity measures in presented in Figure 7.4 in the simple case where we have two classes ($m = 2$).

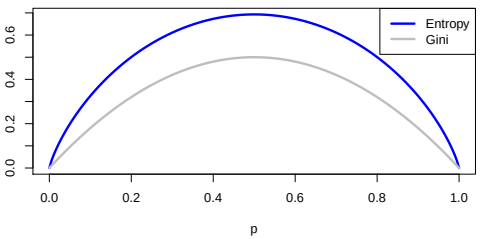


Figure 7.4: Heterogeneity criterions for classification. Both are minimal for $p = 0$ or $p = 1$, and maximal for $p = 1/2$.

4 Pruning the maximal tree

The previous construction leads to a maximal tree $A_{\max}$ (depending on the stopping rule) with K leaves, that is generally very unstable and heavily depends on the training sample: it is overfitted. We have to build a more parsimonious, and hence more robust, prediction model. This will be achieved by *pruning* the maximal tree. We have to find a compromise between the trivial tree reduced to the root (which is underfitted) and the maximal tree $A_{\max}$. The prediction performances of various trees could be compared on a *validation set*. All sub trees of the maximal tree are admissible, but they are generally too many to be all considered. To get around this problem, Breiman et al. (1984)[9] have proposed an algorithm, based on a penalized criterion, to build an *nested sequence of sub-trees* of the maximal tree. One then chooses, among this sequence, the optimal tree minimizing a generalization error.

4.1 Construction of Breiman's subsequence

For a given tree A, we denote by $|A|$ its number of leaves or terminal nodes κ. The value of $|A|$ is a measure of the complexity of the tree A. We define the fit quality of a tree A by

$$D(A) = \sum_{\kappa=1}^{|A|} D_\kappa$$

where D_κ is the heterogeneity of the terminal node κ of the tree A. The construction of Breiman's subsequence relies on the penalized criterion

$$C(A) = D(A) + \gamma \times |A|.$$

For $\gamma = 0$, $A_{\max}$ minimizes $C(A)$. When γ increases, a pruned tree will be preferable to the maximal tree. More precisely, Breiman's subsequence is obtained as follows:

- Let A_1 be the sub tree of $A_{\max}$ (maximal tree) obtained by pruning the nodes κ such that $D(\kappa) = D(\kappa_L) + D(\kappa_R)$.
- For each node in A_1, $D(\kappa) > D(\kappa_L) + D(\kappa_R)$ and $D(\kappa) > D(A_1^\kappa)$ where A_1^κ is the subtree of A_1 from node κ.
- For γ small enough, for all node κ of A_1, $D(\kappa) + \gamma > D(A_1^\kappa) + \gamma |A_1^\kappa|$. This holds while, for all node κ of A_1,

$$\gamma < (D(\kappa) - D(A_1^\kappa))/(|A_1^\kappa| - 1) = s(\kappa, A_1^\kappa).$$

We then define

$$\gamma_1 = \inf_{\kappa \text{ node of } A_1} s(\kappa, A_1^\kappa) = s(\kappa^*, A_1^{\kappa^*}).$$

- $Crit_{\gamma_1}(\kappa^*) = Crit_{\gamma_1}(A_1^{\kappa^*})$ and, for $\gamma = \gamma_1$, the node κ^* becomes preferable to the subtree $A_1^{\kappa^*}$.
- A_2 is the subtree obtained by pruning the branches from the nodes κ^* minimizing $s(\kappa, A_1^\kappa)$: this gives the second tree in the sub-sequence.
- This process is iterated.

We obtain a nested sequence of sub trees

$$A_{\max} \supset A_1 \supset A_2 \supset \cdots A_K$$

where A_K is the trivial tree, reduced to the root, gathering all the training sample.

4.2 Determination of the optimal tree

Once the nested sequence of trees is obtained, we have to determine an optimal one, minimizing the generalization error. As explained in Chapter 2, this error can be estimated on a validation set. More often, V-fold cross-validation is used. In this case, the implementation of the V-fold cross-validation is particular since for each of the V subsamples composed of $V-1$ folds, we obtain a different sequence of trees. In fact, the aim of cross-validation is to determine the optimal value of the penalization parameter γ resulting from Breiman's subsequence produced with the whole training set. We then choose the tree associated with this optimal value of γ. In the cross-validation procedure, for each value of γ produced by Breiman's subsequence, the mean error is computed for the V subtrees . This leads to an optimal value of γ, minimizing the prediction error estimated by cross-validation. We then retain the tree corresponding to this value of γ in Breiman's subsequence.

Algorithm 3 describes the selection of an optimal tree :

Algorithm 3 Selection of an optimal tree by cross-validation

Construction of the maximal tree $A_{\max}$
Construction of Breiman's sequence $A_1 \dots A_K$ of nested trees associated with the
Sequence of penalization parameters $(\gamma_1, \dots, \gamma_K)$
for $v = 1, \dots, V$ **do**
For each sample (composed of $V - 1$ folds), estimation of the sequence of trees associated with the sequence of penalization parameters $(\gamma_K, \dots, \gamma_1)$.
Estimation of the error on the validation fold.
end for
For each value $(\gamma_K, \dots, \gamma_1)$, computation of the means of these errors.
Determination of the optimal value γ_{Opt}, corresponding to the minimal error mean.
Retain the tree corresponding to γ_{Opt} in Breiman's subsequence $A_1, \dots, A_K$.

4.3 Practical remarks

Misclassification cost

For some classification problems, the consequences of misclassification may be more serious from some classes than for others. For example, if tap water is infected by some pollutant dangerous for health, it is worse to predict that the water is drinkable if it is not than vice versa. To account this problem, we define a $m \times m$ loss matrix L (m being the number of classes), where $L_{\ell\ell'}$ denotes the loss incurred for classifying an observation from class ℓ into the class ℓ'. $L_{\ell\ell} = 0$ for all ℓ. For binary classification problems, the loss can be incorporated in the Gini index of cross-entropy by weighting the observations in class ℓ by $L_{\ell\ell'}$. This can also be used for multi-class classification if $L_{\ell\ell'}$ does not depend on ℓ'. In a terminal node κ, we classify to the class minimizing the loss:

$$\hat{k}(\kappa) = \text{argmin}_k \sum_{\ell=1}^{m} L_{\ell k} p_\kappa^\ell.$$

Missing predictor values

CART is tolerant to missing data. For the prediction phase, assume that the dataset has some missing predictor values for some (or all) of the variables. Instead of discarding observations with missing values, or imputing the missing values, CART proposes two better strategies. First, for categorical variables, we can add a category for "missing". The second approach is to construct surrogate variables that will be considered if the value of a variable is missing. We choose as usual the best predictor and split at one node, the first surrogate is the second best, and so on.. When an observation is sent down the tree (during the training or prediction phase), if the value of a predictor is missing at one node, we use the first surrogate; if this one is missing, we use the second and so on.. CART is also tolerant to missing data in the learning phase : the data with missing value for Y or for all the explanatory variables are eliminated,

the decrease in heterogeneity for each variable and split is computed by using the associated available observations and the best split is chosen as usual.

Instability of trees

A major drawback of trees is their high variance. They are not robust, in the sense that a small change in the data can lead to very different sequences of splits. This is why we have to be careful with the interpretation. This is due to the hierarchical procedure : an error in the choice of a split in the top of the tree cannot be corrected below. This instability is the price to pay to have a simple and interpretable model. We will see in Chapter 8 how to aggregate trees to reduce the variance of the prediction rule.

Lack of smoothness

In a regression framework, the trees are constant piecewise functions, they are hence not smooth (not even continuous). This may be a problem if the phenomenon to model is regular. More regular algorithms, such as the MARS procedure have been developed (see Hastie and al [21]).

5 Application to Ozone data

5.1 Regression tree

A regression tree is estimated to predict ozone concentration. The package `rpart` of the software R uses a pruning procedure by cross-validation to optimize the penalty parameter. The tree (see Figure 7.5) recovers the important variables involved in the prediction, but due to the tree structure, this list is not quite similar to the one obtained in a linear model. We see in particular here the complexity of the interaction between the deterministic prediction MOCAGE and the important effect of the temperature in various situations. The residuals on the test sample of the regression tree have a particular structure since the

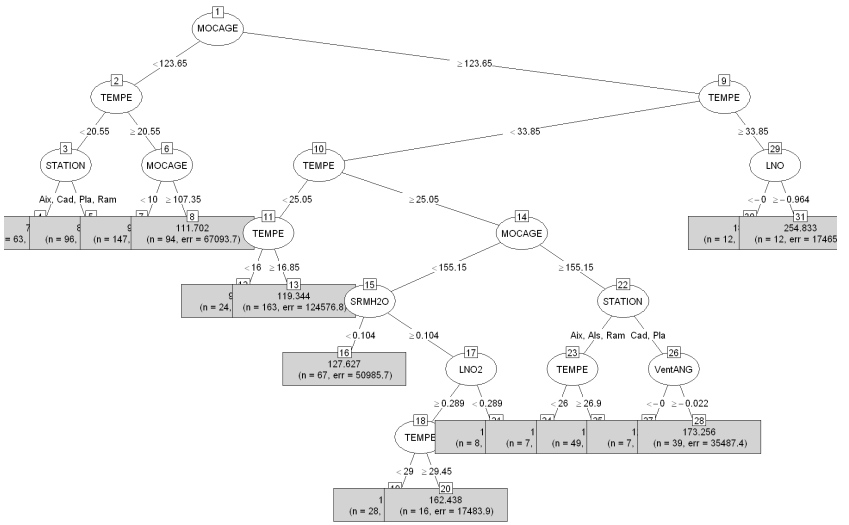


Figure 7.5: *Ozone: regression tree pruned by cross-validation (R).*

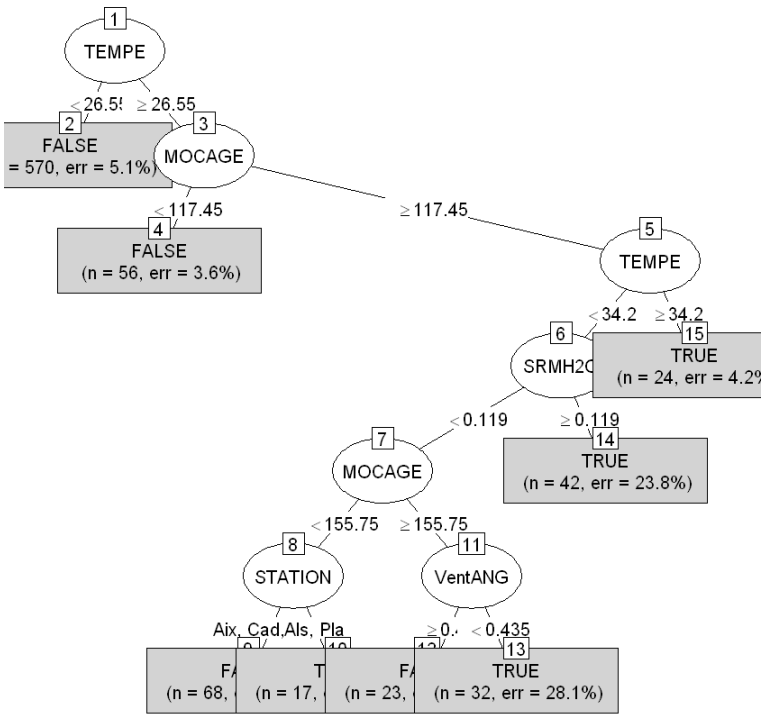


Figure 7.6: *Ozone: classification tree pruned by cross-validation (R).*

same prediction value is obtained for observation falling in the same terminal node. This is why we observe a column per leaf.

5.2 Classification tree

A classification tree is estimated (see Figure 7.6) in order to predict a threshold overflow. It is of comparable complexity with the regression tree, but the variables do not play the same role. The temperature appears here as the "most important" instead of MOCAGE in the regression tree. Confusion matrices exhibit the same biases as regression models by omitting a large number of exceedances.

6 Conclusion

Trees have nice properties : they are easy to interpret, efficient algorithms exist to prune them, they are tolerant to missing data. All these properties made the success of CART for practical applications. Nevertheless, CART algorithm has also important drawbacks : it is highly instable, being not robust to the learning sample and it also suffers from the curse of dimensionality. The selected tree only depends on few explanatory variables, which is nice for the interpretation but trees are often (wrongly) interpreted as a variable selection procedure, due to their high instability. Moreover, prediction accuracy of a tree is often poor compared to other procedures. This is why more robust procedures, based on the aggregation of trees leading to Random Forests have been proposed . They also have better prediction accuracy. This is the topic of Chapter 8.

Chapter 8

Aggregation and Random Forests

1 Introduction

We present in this chapter algorithms based on a random construction of a family of models: bagging for bootstrap aggregating (Breiman 1996) [7] and the random forests of Breiman (2001) [8] which proposes an improvement of *bagging* specific to models defined by binary trees (CART).

The principle of *bagging* applies to any modeling method (regression, CART, neural networks..) but are mostly interesting, and significantly reduces the prediction error, only in the case of *unstable* models. Thus, the use of this algorithm makes little sense with linear regression or discriminant analysis. It is mainly implemented in association with binary trees as a basic models. In fact, the already underlined instability of trees appears as a property favoring the reduction of variance by aggregation of these models.

2 Bagging

2.1 Principle and algorithm

Let Y be a quantitative or qualitative variable, $X^1, \ldots, X^p$ the explanatory variables and $\hat{f}(\mathbf{x})$ a predictor, with $\mathbf{x} = (x^1, \ldots, x^p) \in \mathbb{R}^p$. We denote by n the number of observations and

$$\mathbf{Z} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$$

a sample with distribution F.

Considering B *independent samples* denoted $\{\mathbf{Z}_b\}_{b=1,B}$, a predictor by *model aggregation* is defined below in the case where the variable to explain Y is:

- quantitative : $\widehat{f}_B(.) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_{\mathbf{Z}_b}(.)$,
- qualitative : $\widehat{f}_B(.) = \arg\max_j \text{card}\left\{b \mid \widehat{f}_{\mathbf{Z}_b}(.) = j\right\}$.

In the first case, it is a simple mean of the results obtained for the models associated with each sample, in the second case, a *majority vote*. In the latter

case, if the model returns probabilities associated with each modality as in the logistic regression model, it is simple to calculate these probabilities.

The principle is elementary, averaging the predictions of several independent models allows to *reduce the variance* and therefore to reduce the prediction error.

However, it is unrealistic to consider B independent samples. This would require too much data. These samples are therefore replaced by B *bootstrap* samples each obtained by n draws with replacement according to the empirical measure $\widehat{F}_n$. This leads to the following algorithm.

Algorithm 4 *Bagging*

Let $\mathbf{x}_0$ and
$\mathbf{Z} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ a learning sample.
for $b = 1$ to B **do**
  Draw a bootstrap sample of size n with replacement $\mathbf{z}_b$.
  Estimate $\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ with the bootstrap sample.
end for
Compute the mean $\widehat{f}_B(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B} \widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ or the result of a majority vote.

Figure 8.1 presents two bootstrap samples and the corresponding models built with CART algorithm.

However, the B boostrap samples are built on the same learning sample $\mathbf{Z} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ and therefore the estimators $\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ are not independent. Let us assume that, for all b, $\mathbb{E}(\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)) = f(\mathbf{x}_0)$, $\text{Var}(\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)) = V(\mathbf{x}_0)$ and for all $b \neq b'$, $\text{Corr}(\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0), \widehat{f}_{\mathbf{z}_{b'}}(\mathbf{x}_0)) = \rho(x_0)$.

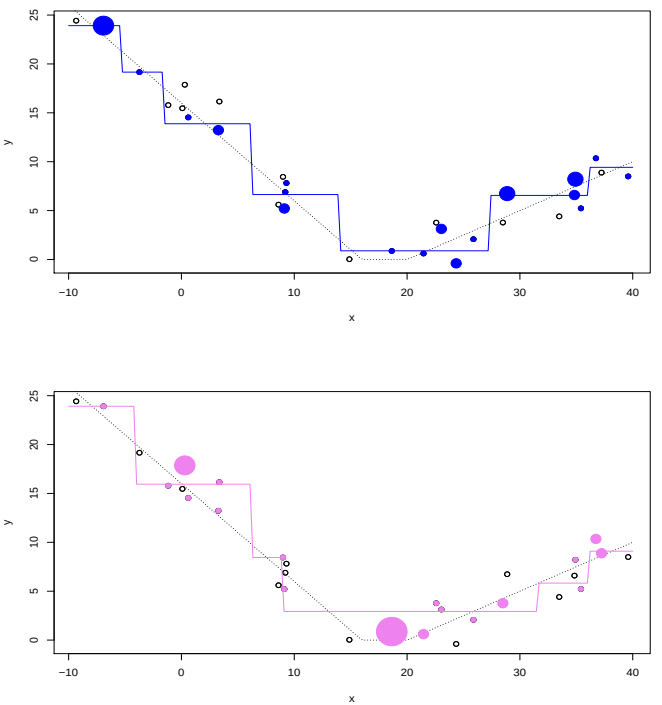


Figure 8.1: Two Bootstrap samples and the two corresponding models built with CART. The point size is proportional to the number of replicates.

Then, in the regression case, we obtain

$$\begin{aligned}\mathbb{E}(\widehat{f}_B(\mathbf{x}_0)) &= f(\mathbf{x}_0)\\ \text{Var}(\widehat{f}_B(\mathbf{x}_0)) &= \rho(x_0)V(\mathbf{x}_0) + \frac{(1-\rho(x_0))}{B}V(\mathbf{x}_0)\\ &\to \rho(x_0)V(\mathbf{x}_0) \text{ as } B \to +\infty\end{aligned}$$

Hence, if the correlation term $\rho(x_0)$ is small, the variance of the aggregated predictor $\widehat{f}_B(\mathbf{x}_0)$ is much smaller than the one of a single predictor. This underlines the importance of finding low correlated predictors $(\widehat{f}_b(\mathbf{x}_0))_{1\leq b\leq B}$, which is at the core of the **Random forests** algorithm.

3 Random Forests

3.1 Motivation

In the specific case of binary decision tree models (CART), Breiman (2001) [8] proposes an improvement of the *bagging* by adding a random component. The objective is to make the aggregated trees more *independent* by adding randomness in the choice of the variables which are involved in the prediction. Since the initial publication of the algorithm, this method has been widely tested and compared with other procedures see Fernandez-Delgado et al. 2014 [16], Caruana et al. 2008 [10]. It becomes in many machine learning articles the method to beat in terms of prediction accuracy. Theoretical convergence properties, difficult to study, have been published quite recently (Scornet et al. 2015) [36]. However, it can also lead to bad results, especially when the underlying problem is linear.

3.2 Algorithm

The *bagging* is applied to binary decision trees by adding a random selection of m explanatory variables among the p variables.

Algorithm 5 Random Forests

Let $\mathbf{x}_0$ and
$\mathbf{Z} = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ a learning sample
for $b = 1$ to B **do**
  Take a bootstrap sample $\mathbf{z}_b$
  Estimate a tree on this sample with **randomization** of the variables : the search for each optimal division is preceded by a random selection of a subset of m predictors.
end for
Calculate the mean estimate $\widehat{f}_B(\mathbf{x}_0) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}_{\mathbf{z}_b}(\mathbf{x}_0)$ or the result of a majority vote.

Parameters of the algorithm

The pruning strategy can, in the case of random forests, be quite elementary. Indeed, pruned trees may be strongly correlated because they may involve the same variables appearing to be the most explanatory. In the default strategy of the algorithm, it is simply the minimum number of observations per leaf which limits the size of the tree, it is set to 5 by default. We therefore aggregate rather complete trees, which are considered of low bias but of high variance.

The random selection of a reduced number of m potential predictors at each stage of the construction of the trees significantly increases the variability by highlighting other variables. Each tree is obviously less efficient, sub-optimal, but, united being strength, aggregation ultimately leads to good results. The number m of variables drawn randomly can, according to the examples, be a sensitive parameter with default choices are not always optimal :

- $m = \sqrt{p}$ in a classification problem,
- $m = p/3$ in a regression problem.

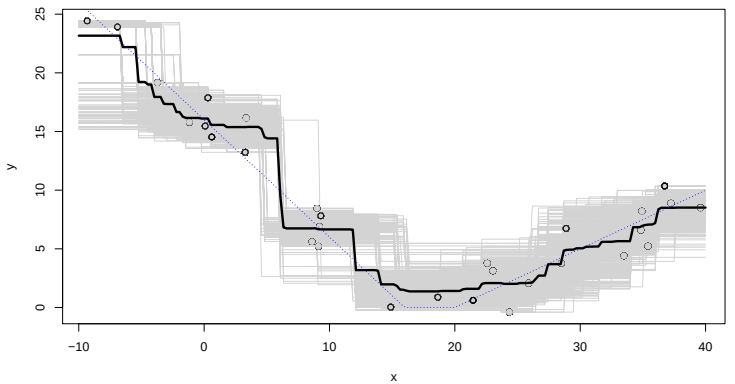


Figure 8.2: 500 bootstrap samples (grey), corresp. predictions with tree, and their average (bold line). The function to be estimated in dotted blue line.

The iterative evaluation of the *out-of-bag* error makes it possible to control the number B of trees in the forest as well as to optimize the choice of m. It is nevertheless a cross-validation procedure which is preferably used to optimize m. The Figure 8.2 presents an example of Random Forest regression predictor, built with $B = 500$ bootstrap samples.

3.3 Variables importance

Random forests generally have a good accuracy, they are easily implementable, parallelisable but not easy to interpret like any model built by aggregation, leading to a black-box model. To favor interpretation, indexes of importance for each explanatory variable have been introduced. This is obviously all the more useful as the variables are very numerous. Two criteria have been proposed to evaluate the importance of the variable X^j.

- The first one Mean Decrease Accuracy (MDA) is based on a random permutation of the values of this variable. The more the quality of the prediction, estimated by an out-of-bag error, is degraded by the permutation of this variable, the more the variable is important. Once the bth tree has been constructed, the *out-of-bag* sample is predicted for this tree and the estimated error is recorded. The values of the jth variable are then randomly permuted in the out-of-bag data sample and the error is computed again. The decrease in prediction accuracy is averaged over all the trees and used to assess the importance of the variable X^j in the forest. It is therefore a global but indirect measure of the influence of a variable on the quality of forecasts. More formally,
  - Consider a variable X^j and denote by $\mathcal{D}_{b,n}$ the out-of-bag data set of the b-th tree and $\mathcal{D}^j_{b,n}$ the same data set where the values of X^j have been randomly permuted.
  - Denote by $\widehat{f}_{\mathbf{z}_b}$ the b-th tree estimate and

$$R_n[\widehat{f}_{\mathbf{z}_b}, \mathcal{D}] = \frac{1}{|\mathcal{D}|} \sum_{i,(X_i,Y_i)\in\mathcal{D}} (Y_i - \widehat{f}_{\mathbf{z}_b}(X_i))^2.$$

  - The MDA is defined by

$$\text{MDA}(X^j) = \frac{1}{B} \sum_{b=1}^{B} \{R_n[\widehat{f}_{\mathbf{z}_b}, \mathcal{D}^j_{b,n}] - R_n[\widehat{f}_{\mathbf{z}_b}, \mathcal{D}_{b,n}]\}.$$

- The second variable importance criterion is the Mean Decrease Impurity (MDI). It is a is local criterion, based on the average of the decrease of

heterogeneity each time the variable X^j is chosen as a split at some node. More formally, with previous notations, the MDI of the variable X^j is defined by

$$\text{MDI}(X^j) = \frac{1}{Bn}\sum_{b=1}^{B}\sum_{\kappa\in\mathcal{T}_b, j_\kappa^\star = j}[D_\kappa - (D_{\kappa_L}(t_\kappa^\star, j_\kappa^\star) + D_{\kappa_R}(t_\kappa^\star, j_\kappa^\star))],$$

- $\{\mathcal{T}_b, 1 \leq b \leq B\}$ is the collection of trees in the forest,
- $(t_\kappa^\star, j_\kappa^\star)$ the split retained at node κ :
  - $j_\kappa^\star$ corresponds to the optimal variable selected for the split
  - $t_\kappa^\star$ corresponds to the optimal threshold along the $j_\kappa^\star$ variable.

Example on ozone data :

Details (from R help file of function `importance`)
"The first measure [%IncMSE] is computed from permuting OOB data : For each tree, the prediction error on the out-of-bag portion of the data is recorded (error rate for classification, MSE for regression). Then the same is done after permuting each predictor variable. The difference between the two are then averaged over all trees, and normalized by the standard deviation of the differences.
The second measure [IncNodePurity] is the total decrease in node impurities from splitting on the variable, averaged over all trees. For classification, the node impurity is measured by the Gini index. For regression, it is measured by residual sum of squares."

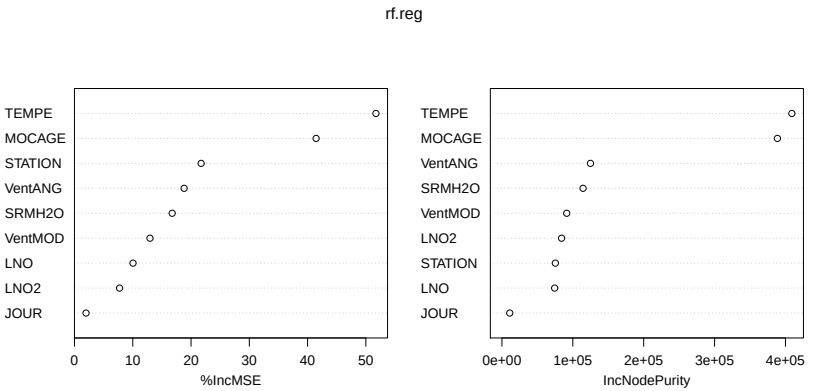


Figure 8.3: Variable importance plot, returned by the R function `importance`. MDA on the left and MDI on the right.

3.4 Implementation

- The `randomForest` library of R interfaces the original program developed in Fortran77 by Leo Breiman and Adele Cutler which maintains the site dedicated to this algorithm.

- An alternative in R, more efficient in computing time especially with a large volume of data, consists in using the `ranger` library.
- The software site Weka developed at Waikato University in New Zealand offers a version in Java.
- A very efficient and close version of the original algorithm is available in the `Scikit-learn` library of Python.
- Another version suitable for big data is available in the `MLlib` library of Spark, a technology developed to interface different hardware/-software architectures with distributed data file management systems (Hadoop). In addition to the usual parameters : number of trees, maximum depth of trees, and number of variables drawn at random to build a subdivision at each node, this implementation adds two parameters : `subsamplingRate` and `maxBins`, which have a default value. These parameters play an important role, certainly in drastically reducing the computation time, but, on the other hand, in restricting the precision of the estimate. They regulate the balance between computation time and precision of the estimate as a subsampling in the data would do.
  - **subsamplingRate =1.0** subsamples as its name suggests before building each tree. With the default value, it is the classic version of random forests with B Bootstrap samples of size n but if this rate is less than 1, smaller samples are drawn. The sample are then more distinct (or independent) for each tree. The variance is therefore reduced (more independent trees) but the bias increases because each tree is built with a smaller data set.
  - **maxBins = 32** is the maximum number of categories that are considered for a qualitative variable or the number of possible values for a quantitative variable. Only the most frequent modalities of a qualitative variable are taken into account, the others are automatically grouped into a `other` modality. As previously, the time to determine a better division is obviously largely influenced by the number of modalities or even the number of possible values of a quantitative variable. Reducing the number of possible values is finally another way of reducing the computation time but it would be appropriate to guide the groupings of the modalities to avoid misinterpretations.

4 Conclusion

Having become the *Swiss Army Knife* of learning, Random Forests are used for different purposes (see the dedicated site) :

- Similarity or proximity between observations : after building each tree, increment by 1 the similarity or proximity of two observations that are in the same leaf. Sum on the trees of the forest, normalize by the number of trees. A multidimensional positioning can represent these similarities or the matrix of dissimilarities that results from them.
- Detection of multidimensional atypical observations : *outliers* or *novelties* that correspond to observations which do not belong to known classes. A criterion of "abnormality" with respect to a class is based on the previous notion of proximities of an observation to the other observations of its class.
- Another algorithm, inspired by Random Forests has been developed for anomaly detection, it is called *isolation forest*.
- Random forests are used for the Imputation of missing data.
- Adaptations to take into account censored data to model survival times correspond to the *survival forest* algorithm.

Chapter 9

Aggregation by boosting algorithms

1 Introduction

Boosting was originally introduced for binary classification problems but it has been also extended to k class classification and to regression problems. The main idea is to combine weak classifiers (or regressors) in a way to get a powerful aggregated procedure. The aggregation allows to reduce the variance of the single predictors but also their bias (which is a fundamental difference with the bagging which does not modify the bias). The final prediction rule is obtained as a linear combination of a recursive sequence of predictors, where each predictor is an adaptive version of the previous one, given more weight, in the next estimator, to the observations that are badly adjusted at the previous step. We first present the most popular boosting algorithm, introduced by Freund and Schapire [17], called AdaBoost for binary classification.

2 AdaBoost algorithm

Adaboost (or adaptive boosting) is a boosting method introduced by Freund and Schapire [17] to combine several classifiers $f_1, \ldots, f_p$. It is devoted to binary classification where the output $Y \in \{-1, 1\}$. The principle of the algorithm is to minimize the empirical risk for the exponential loss function over the linear space $\mathcal{F}$ generated by the classifiers $f_1, \ldots, f_p$. The aim is to compute

$$\hat{f} = \underset{f \in \text{span}(f_1, \ldots, f_p)}{\text{argmin}} \{\frac{1}{n} \sum_{i=1}^{n} \exp(-Y_i f(X_i))\}. \tag{9.1}$$

Let us recall that it is convenient to consider a convex loss function ℓ satisfying the condition $\ell(z) \geq \mathbf{1}_{z<0}$, which allows to give an upper bound for the misclassification probability for the prediction rule given by the sign of $f(X)$. Indeed

$$\mathbb{E}(\ell(Yf(X))) \geq \mathbb{E}(\mathbf{1}_{Yf(X)<0}) = \mathbb{P}(Y \neq sign(f(X))).$$

Classical convex losses ℓ are the hinge loss $\ell(z) = (1-z)_+$, (used for SVM's for example), the exponential loss $\ell(z) = \exp(-z)$, the logit loss $\ell(z) = \log_2(1+\exp(-z))$.
Since this optimization problem (9.1) is complex, in order to approximate the solution, Adaboost computes a recursive sequence of predictors $\hat{f}_m$ for $m = 0, \dots M$ with

$$\begin{aligned} \hat{f}_0 &= 0 \\ \hat{f}_m &= \hat{f}_{m-1} + \beta_m f_{j_m} \end{aligned}$$

where (β_m, j_m) minimizes the empirical risk associated to the exponential loss function:

$$(\beta_m, j_m) = \underset{\beta\in\mathbb{R}, j=1,\dots,p}{\operatorname{argmin}} \{\frac{1}{n}\sum_{i=1}^{n} \exp(-Y_i(\hat{f}_{m-1}(X_i) + \beta f_j(X_i)))\}. \quad (9.2)$$

Hence, at each step, the algorithm looks for the best classifier f_{j_m} in the available collection $f_1, \dots, f_p$ and the best coefficient β_m such that adding $\beta_m f_{j_m}$ to the previous predictor $\hat{f}_{m-1}$ minimizes the empirical exponential loss.
The final classification rule is given by

$$\hat{f} = \operatorname{sign}(\hat{f}_M).$$

M is a parameter of the procedure.
The aim of the following exercise is to compute the solution of (9.2).

Exercise. — We denote

$$w_i^{(m)} = \frac{1}{n}\exp(-Y_i \hat{f}_{m-1}(X_i))$$

and we assume that for all $j = 1, \dots, p$,

$$err_m(j) = \frac{\sum_{i=1}^n w_i^{(m)} \mathbf{1}_{sign(f_j(X_i)) \neq Y_i}}{\sum_{i=1}^n w_i^{(m)}} \in]0,1[.$$

Prove that

$$j_m = \underset{j=1,\dots,p}{\operatorname{argmin}} err_m(j),$$

and

$$\beta_m = \frac{1}{2}\log\left(\frac{1-err_m(j)}{err_m(j)}\right).$$

Note that the initial weights are equal for all the observations : $w_i^{(1)} = 1/n$ for $i = 1, \dots, n$ and that the AdaBoost algorithm then attributes more weights in the computation of the predictor at step m to the observations for which the exponential loss $\exp(-Y_i \hat{f}_{m-1}(X_i))$ of the previous estimator is high.

This leads to the AdaBoost algorithm :

Algorithm 6 AdaBoost

Choose a parameter M.
$w_i^{(1)} = 1/n$ for $i = 1, \dots, n$.
for $m = 1, \dots, M$ **do**
  $j_m = \underset{j=1,\dots,p}{\operatorname{argmin}} err_m(j)$
  $\beta_m = \frac{1}{2}\log\left(\frac{1-err_m(j)}{err_m(j)}\right)$
  $w_i^{(m+1)} = w_i^{(m)} \exp(-Y_i \beta_m f_{j_m}(X_i))$ for $i = 1, \dots, n$.
end for
$\hat{f}_M(x) = \sum_{m=1}^M \beta_m f_{j_m}(x)$.
$\hat{f} = \operatorname{sign}(\hat{f}_M)$.

The introduction of the exponential loss is motivated by computational reasons in the context of a sequentially additive modeling approach: it leads to the simple reweighting AdaBoost algorithm. One can wonder about the relevance of this exponential loss function. The aim of the following exercise is to

show that minimizing the exponential loss (at the population level) leads to the Bayes classifier. Although this is not true for the empirical loss, this justifies the use of the exponential loss.

Exercise. — Let

$$f^*(x) = \underset{f}{\text{argmin}}\mathbb{E}\left(e^{-Yf(X)}/X = x\right).$$

Prove that

$$f^*(x) = \frac{1}{2}\log\left(\frac{\mathbb{P}(Y=1/X=x)}{\mathbb{P}(Y=-1/X=x)}\right).$$

Deduce that $sign(f^*(x))$ corresponds to the Bayes classifier.

3 Boosting as stagewise additive modelling

The boosting relies on an additive model based on a set of elementary basis functions of the form

$$\hat{f}_M(x) = \sum_{m=1}^{M} \beta_m b(x, \gamma_m),$$

where $(\beta_m)_{1\leq m\leq M}$ are real coefficients and $x \mapsto b(x,\gamma)$ is a function of the multivariate input variable x, depending on a set of parameters γ. This can be a weak classifier for classification problem, a quite simple tree for regression or classification purposes, a perceptron with a single hidden layer ... Given a loss function ℓ, the goal is to solve the following optimization problem:

$$\min_{\{\beta_m,\gamma_m\}_{m=1}^M} \sum_{i=1}^{n} \ell(Y_i, \sum_{m=1}^{M} \beta_m b(X_i, \gamma_m)). \tag{9.3}$$

For classification purposes, the loss function ℓ can be the exponential loss, the cross-entropy (or deviance), or the logit lost. In the regression case, we may consider the $\mathbb{L}^2$ loss, leading to the $\mathbb{L}^2$ boosting. Sometimes the $\mathbb{L}^1$ loss is prefered since it is more robust (less sensitive to outliers). The Huber loss combines the advantages of the $\mathbb{L}^2$ loss (it is differentiable) and the $\mathbb{L}^1$ loss (robustness). It is defined by

$$\ell(y,y') = (y-y')^2\mathbf{1}_{|y-y'|\leq\delta} + (2\delta|y-y'| - \delta^2)\mathbf{1}_{|y-y'|>\delta},$$

where δ is a positive parameter.

The optimization problem (9.3) being generally complex to solve, a step by step additive procedure allows to approximate the solution. This leads to the stagewise additive model, where we solve at each step

$$\begin{aligned}\hat{f}_0 &= 0\\ \hat{f}_m &= \hat{f}_{m-1} + \beta_m b(x,\gamma_m)\end{aligned}$$

with

$$(\beta_m,\gamma_m) = \underset{(\beta,\gamma)}{\text{argmin}}\{\frac{1}{n}\sum_{i=1}^{n}\ell(Y_i, \hat{f}_{m-1}(X_i) + \beta b(X_i,\gamma))\}. \tag{9.4}$$

The predictor $\hat{f}_m(x)$ is an improvement of the one obtained at the previous step. AdaBoost is a particular case of stagewise adaptive modeling with the exponential loss. Let us now consider the particular case of trees.

A tree with J leaves can be expressed as

$$b(x,\gamma) = \sum_{j=1}^{J} \lambda_j \mathbf{1}_{x\in R_j},$$

with $\gamma = (\{R_j, \lambda_j\}, j = 1 \dots J)$.
A boosting tree model is a sum of such trees

$$\hat{f}_M(x) = \sum_{m=1}^{M} b(x, \gamma_m)$$

where $\hat{f}_0 = 0$, and we solve at step m :

$$\gamma_m = \underset{\gamma}{\text{argmin}}\{\frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, \hat{f}_{m-1}(X_i) + b(X_i, \gamma))\}, \tag{9.5}$$

$\gamma_m = (\{R_{j,m}, \lambda_{j,m}\}, j = 1 \dots J)$ are the region sets and constants corresponding to the next tree. This problem is generally hard to solve but in some cases, it simplifies. In particular, if we consider a regression problem with the $\mathbb{L}^2$ loss, then one can notice that this problem is equivalent to finding a single tree to predict the residuals $Y_i - f_{m-1}(X_i)$.

4 Gradient Boosting Models (GBM)

In the same spirit of stagewise additive modeling, Friedman [18] proposed and algorithm called MART for *Multiple Additive Regression Trees*, generalized as Gradient Boosting Models. This algorithm is based on a differentiable loss function L. The principle is still to construct a stagewise additive model. Each model, added to the linear combination, is a step to a better solution. The main innovation is that this step is done in the direction of the gradient of the loss function (assumed to be differentiable) , this gradient being itself approximated by a regression tree. The algorithm is described below in the regression case. It can also be adapted for classification.

Note that, in the case of the $\mathbb{L}^2$ loss, setting $\ell(y, y') = \frac{1}{2}(y - y')^2$, we have

$$-r_{im} = Y_i - f_{m-1}(X_i),$$

Algorithm 7 *Gradient Tree Boosting* for regression

Initialize $\widehat{f}_0(x) = \arg\min_\gamma \sum_{i=1}^n \ell(Y_i, \gamma)$ for all x.
for $m = 1$ to M **do**
Compute $r_{im} = -\left[\frac{\partial \ell(Y_i, f(X_i))}{\partial f(X_i)}\right]_{f=f_{m-1}}; i = 1, \dots, n$
Adjust a regression tree δ_m to the data $(X_i, r_{im})_{i=1,\dots,n}$. The terminal regions are denoted $(R_{jm}, j = 1, \dots J_m)$.
Compute $\gamma_{jm} = \underset{\gamma}{\text{argmin}} \sum_{X_i \in R_{jm}} \ell(Y_i, f_{m-1}(X_i) + \gamma)$.
Update $\widehat{f}_m(x) = \widehat{f}_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{x \in R_{jm}}$.
end for
Output $\widehat{f}(x) = \widehat{f}_M(x)$.

which means that the negative gradient is simply equal to the residuals. In this case, the Gradient tree boosting is equivalent to the $\mathbb{L}^2$ boosting trees presented in the previous section. For other losses, such that the Huber loss, this leads to new procedures. In a classification problem, we can consider for the loss function ℓ the deviance or the exponential loss.
Generally, the size J_m of the tree at step m is taken to be constant for all m. This constant size J is a parameter of the procedure.

5 Regularization

For boosting procedures, the number of iterations M has to be calibrated. Each iteration reduces the training error, hence, large values of M lead to overfitting. A convenient manner to calibrate this parameter is to estimate the generalization error on a validation sample, as a function of M and to choose the value of M minimizing this validation error (this procedure is called *early stopping*.

Another way to avoid overfitting is to use a shrinkage method. In the case of boosting procedures, this corresponds to scaling the contribution of each tree by a factor $0 < \nu < 1$, which leads to the formula

$$\widehat{f}_m(x) = \widehat{f}_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{x \in R_{jm}}$$

in Algorithm 7. This parameter ν corresponds to the learning rate of the gradient descent procedure. Reducing the value of ν increases the training error, and increases the number of iterations M. It seems that the best strategy is to choose a small value for ν ($\nu < 0.1$) and to calibrate M by early stopping, even if this has a computational cost. Generally, small trees (small value of J) with no pruning are used at each step of the boosting algorithm, which limits the computational cost.

It is also common to use *subsampling*, leading to a *Stochastic Gradient Boosting* procedure. At each iteration, a fraction η of the training sample is drawn without replacement, and only this subsample is used to estimate the next tree. This allows to reduce the computing time but also produces more accurate models, since, like for the bagging, the successive trees are less dependent, which allows to reduce the variance of the final estimator. Note that this procedure has also common points with batch learning for neural networks.

6 Conclusion

To summarize, the boosting allows to reduce the variance compared with single procedures but also the bias by aggregation, which generally leads to very performant procedures.
Trees are easily interpretable. Of course, the interpretation is lost by aggregating. Nevertheless, like for Random Forest, indices of importance can be computed : one can average over the M trees of the boosting algorithm the indices of importance computed for each tree. This is crucial to have an idea of the relative importance of each predictor in the model.
The Gradient Boosting is implemented in the R package `gbm`. It is also implemented in the Scikit Learn library of Python (GradientBoostingClassifier and GradientBoostingRegressor).

Chapter 10

Neural Networks and Introduction to Deep Learning

1 Introduction

Deep learning is a set of learning methods attempting to model data with complex architectures combining different non-linear transformations. The elementary bricks of deep learning are the neural networks, that are combined to form the deep neural networks.

These techniques have enabled significant progress in the fields of sound and image processing, including facial recognition, speech recognition, computer vision, automated language processing, text classification (for example spam recognition). Potential applications are very numerous. A spectacularly example is the AlphaGo program, which learned to play the go game by the deep learning method, and beated the world champion in 2016.

There exist several types of architectures for neural networks:

- The multilayer perceptrons, that are the oldest and simplest ones
- The Convolutional Neural Networks (CNN), particularly adapted for image processing
- The recurrent neural networks, used for sequential data such as text or times series.

They are based on deep cascade of layers. They need clever stochastic optimization algorithms, and initialization, and also a clever choice of the structure. They lead to very impressive results, although very few theoretical fondations are available till now.

2 Neural networks

An artificial neural network is an application, non linear with respect to its parameters θ that associates to an entry x an output $y = f(x, \theta)$. For the sake of simplicity, we assume that y is unidimensional, but it could also be multidimensional. This application f has a particular form that we will precise. The neural networks can be use for regression or classification. As usual in statistical learning, the parameters θ are estimated from a learning sample. The function to minimize is not convex, leading to local minimizers. The success

of the method came from a universal approximation theorem due to Cybenko (1989) and Hornik (1991). Moreover, Le Cun (1986) proposed an efficient way to compute the gradient of a neural network, called backpropagation of the gradient, that allows to obtain a local minimizer of the quadratic criterion easily.

2.1 Artificial Neuron

An artificial neuron is a function f_j of the input $x = (x_1, \ldots, x_d)$ weighted by a vector of connection weights $w_j = (w_{j,1}, \ldots, w_{j,d})$, completed by a neuron bias b_j, and associated to an activation function ϕ, namely

$$y_j = f_j(x) = \phi(\langle w_j, x\rangle + b_j).$$

Several activation functions can be considered.

- The identity function
$$\phi(x) = x.$$
- The sigmoid function (or logistic)
$$\phi(x) = \frac{1}{1+\exp(-x)}.$$
- The hyperbolic tangent function ("tanh")
$$\phi(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} = \frac{\exp(2x)-1}{\exp(2x)+1}.$$
- The hard threshold function
$$\phi_\beta(x) = \mathbf{1}_{x\geq\beta}.$$
- The Rectified Linear Unit (ReLU) activation function
$$\phi(x) = \max(0, x).$$

Here is a schematic representation of an artificial neuron where $\Sigma = \langle w_j, x\rangle + b_j$. The Figure 10.2 represents the activation function described above. Historically, the sigmoid was the mostly used activation function since it is differentiable and allows to keep values in the interval $[0, 1]$. Nevertheless, it is problematic since its gradient is very close to 0 when $|x|$ is not close to 0. The Figure 10.3 represents the Sigmoid function and its derivative. With neural networks with a high number of layers (which is the case for deep learning), this causes troubles for the backpropagation algorithm to estimate the parameter (backpropagation is explained in the following). This is why the sigmoid function was supplanted by the rectified linear function. This function is not differentiable in 0 but in practice this is not really a problem since the probability to have an entry equal to 0 is generally null. The ReLU function also has a sparsification effect. The ReLU function and its derivative are equal to 0 for

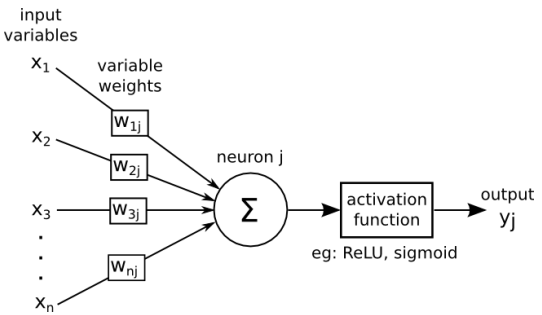


Figure 10.1: source: andrewjames turner.co.uk

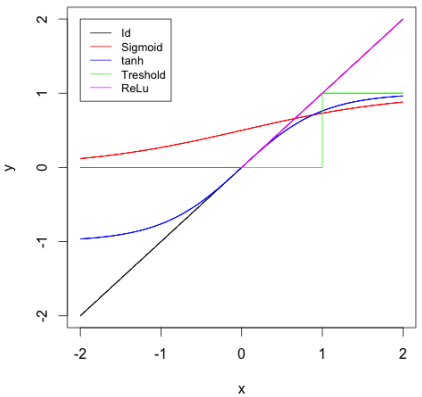


Figure 10.2: Activation functions

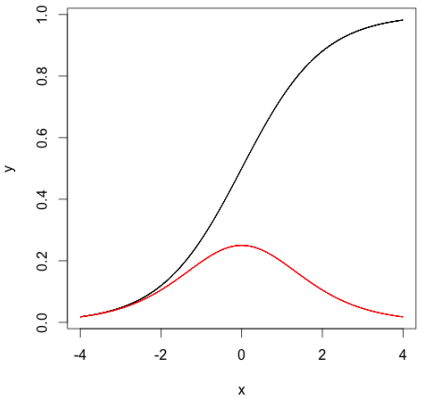


Figure 10.3: Sigmoid function (in black) and its derivatives (in red)

negative values, and no information can be obtain in this case for such a unit, this is why it is advised to add a small positive bias to ensure that each unit is active. Several variations of the ReLU function are considered to make sure that all units have a non vanishing gradient and that for $x < 0$ the derivative is not equal to 0. Namely

$$\phi(x) = \max(x, 0) + \alpha \min(x, 0)$$

where α is either a fixed parameter set to a small positive value, or a parameter to estimate.

2.2 Multilayer perceptron

A multilayer perceptron (or neural network) is a structure composed by several hidden layers of neurons where the output of a neuron of a layer becomes the input of a neuron of the next layer. Moreover, the output of a neuron can also be the input of a neuron of the same layer or of neuron of previous layers (this is the case for recurrent neural networks). On last layer, called output layer, we may apply a different activation function as for the hidden layers depending on the type of problems we have at hand: regression or classification. The Figure 10.4 represents a neural network with three input variables, one output variable, and two hidden layers. Multilayers perceptrons have a basic

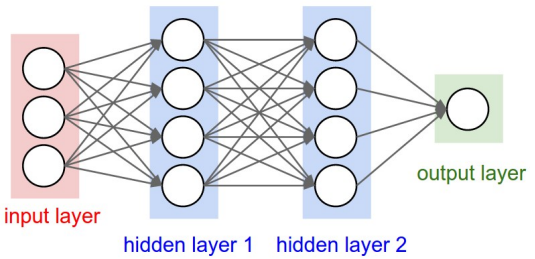


Figure 10.4: A basic neural network. Source: http://blog.christianperone.com

architecture since each unit (or neuron) of a layer is linked to all the units of the next layer but has no link with the neurons of the same layer. The parameters of the architecture are the number of hidden layers and of neurons in each layer. The activation functions need also to be chosen by the user. For the output layer, as mentioned previously, the activation function is generally different from the one used on the hidden layers. In the case of regression, we apply no activation function on the output layer. For binary classification, the output gives a prediction of $\mathbb{P}(Y = 1/X)$ since this value is in $[0, 1]$, the sigmoid activation function is generally considered. For multi-class classification, the output layer contains one neuron per class i, giving a prediction of $\mathbb{P}(Y = i/X)$. The sum of all these values has to be equal to 1. The multidimensional function *softmax* is generally used

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

Let us summarize the mathematical formulation of a multilayer perceptron with L hidden layers.
We set $h^{(0)}(x) = x$.

For $k = 1, \ldots, L$ (hidden layers),

$$\begin{aligned} a^{(k)}(x) &= b^{(k)} + W^{(k)} h^{(k-1)}(x) \\ h^{(k)}(x) &= \phi(a^{(k)}(x)) \end{aligned}$$

For $k = L + 1$ (output layer),

$$\begin{aligned} a^{(L+1)}(x) &= b^{(L+1)} + W^{(L+1)} h^{(L)}(x) \\ h^{(L+1)}(x) &= \psi(a^{(L+1)}(x)) := f(x, \theta). \end{aligned}$$

where ϕ is the activation function and ψ is the output layer activation function (for example softmax for multiclass classification). At each step, $W^{(k)}$ is a matrix with number of rows the number of neurons in the layer k and number of columns the number of neurons in the layer $k - 1$.

2.3 Universal approximation theorem

Hornik (1991) showed that any bounded and regular function $\mathbb{R}^d \to \mathbb{R}$ can be approximated at any given precision by a neural network with one hidden layer containing a finite number of neurons, having the same activation function, and one linear output neuron. This result was earlier proved by Cybenko (1989) in the particular case of the sigmoid activation function. More precisely, Hornik's theorem can be stated as follows.

THEOREM 6. — *Let ϕ be a bounded, continuous and non decreasing (activation) function. Let K_d be some compact set in $\mathbb{R}^d$ and $\mathcal{C}(K_d)$ the set of continuous functions on K_d. Let $f \in \mathcal{C}(K_d)$. Then for all $\varepsilon > 0$, there exists $N \in \mathbb{N}$, real numbers v_i, b_i and $\mathbb{R}^d$-vectors w_i such that, if we define*

$$F(x) = \sum_{i=1}^{N} v_i \phi(\langle w_i, x \rangle + b_i)$$

then we have

$$\forall x \in K_d, |F(x) - f(x)| \leq \varepsilon.$$

This theorem is interesting from a theoretical point of view. From a practical point of view, this is not really useful since the number of neurons in the hidden layer may be very large. The strength of deep learning lies in the deep (number of hidden layers) of the networks.

3 Estimation of the parameters

Once the architecture of the network has been chosen, the parameters (the weights w_j and biases b_j) have to be estimated from a learning sample. As usual, the estimation is obtained by minimizing a loss function with a gradient descent algorithm. We first have to choose the loss function.

Loss functions

It is classical to estimate the parameters by maximizing the likelihood (or equivalently the logarithm of the likelihood). This corresponds to the minimization of the loss function which is the opposite of the log likelihood. Denoting θ the vector of parameters to estimate, we consider the expected loss function

$$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}(\log(p_\theta(Y/X)).$$

If the model is Gaussian, namely if $p_\theta(Y/X = x) \sim \mathcal{N}(f(x,\theta), I)$, maximizing the likelihood is equivalent to minimize the quadratic loss, hence

$$L(\theta) = \mathbb{E}_{(X,Y)\sim P}(\|Y - f(X,\theta)\|^2).$$

For binary classification, with $Y \in \{0,1\}$, maximizing the log-likelihood corresponds to the minimization of the cross-entropy.
Setting $f(X,\theta) = \mathbb{P}_\theta(Y = 1/X)$, the cross-entropy loss is

$$\ell(f(x,\theta), y) = -[y\log(f(x,\theta)) + (1-y)\log(1-f(x,\theta))]$$

and the corresponding expected loss is

$$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}[Y\log(f(X,\theta)) + (1-Y)\log(1-f(X,\theta))].$$

This loss function is well adapted with the sigmoid activation function since the use of the logarithm avoids to have too small values for the gradient.
Finally, for a multi-class classification problem, we consider a generalization of the previous loss function to k classes

$$L(\theta) = -\mathbb{E}_{(X,Y)\sim P}[\sum_{j=1}^{k} \mathbf{1}_{Y=j}\log p_\theta(Y = j/X)].$$

Ideally we would like to minimize the classification error, but it is not smooth, this is why we consider the cross-entropy (or eventually a convex surrogate).

3.1 Penalized empirical risk

The expected loss can be written as $L(\theta) = \mathbb{E}_{(X,Y)\sim P}[\ell(Y, f(X,\theta))]$ and it is associated to a loss function ℓ.
In order to estimate the parameters θ, we use a training sample $(X_i, Y_i)_{1\le i\le n}$ and we minimize the empirical loss

$$\tilde{L}_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f(X_i,\theta))$$

eventually we add a regularization term. This leads to minimize the penalized empirical risk

$$L_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(Y_i, f(X_i,\theta)) + \lambda\Omega(\theta).$$

We can consider $\mathbb{L}^2$ regularization. Using the same notations as in Section 2.2,

$$\begin{aligned}\Omega(\theta) &= \sum_k\sum_i\sum_j (W_{i,j}^{(k)})^2 \\ &= \sum_k \|W^{(k)}\|_F^2\end{aligned}$$

where $\|W\|_F$ denotes the Frobenius norm of the matrix W. Note that only the weights are penalized, the biases are not penalized. It is easy to compute the gradient of $\Omega(\theta)$:

$$\bigtriangledown_{W^{(k)}}\Omega(\theta) = 2W^{(k)}.$$

One can also consider $\mathbb{L}^1$ regularization, leading to parcimonious solutions:

$$\Omega(\theta) = \sum_k \sum_i \sum_j |W_{i,j}^{(k)}|.$$

In order to minimize the criterion $L_n(\theta)$, a **stochastic gradient descent algorithm** is used. In order to compute the gradient, a clever method, called **Backpropagation algorithm** is considered. It has been introduced by Rumelhart et al. (1988), it is still crucial for deep learning.
The stochastic gradient descent algorithm performs at follows:

- Initialization of $\theta_0 = (W^{(1)}, b^{(1)}, \ldots, W^{(L+1)}, b^{(L+1)})$.
- At each iteration, we compute :

$$\theta_j = \theta_{j-1} - \varepsilon \frac{1}{m} \sum_{i \in B} [\bigtriangledown_\theta \ell(f(X_i, \theta_{j-1}), Y_i) + \lambda \bigtriangledown_\theta \Omega(\theta_{j-1})].$$

Note that, in the previous algorithm, we do not compute the gradient for the loss function at each step of the algorithm but only on a subset B of cardinality m (called a *batch*). This is what is classically done for big data sets (and for deep learning) or for sequential data. B is taken at random without replacement. An iteration over all the training examples is called an **epoch**. The numbers of epochs to consider is a parameter of the deep learning algorithms. The total number of iterations equals the number of epochs times the sample size n divided by m, the size of a batch. This procedure is called *batch learning*, sometimes, one also takes batches of size 1, reduced to a single training example $B = \{(X_i, Y_i)\}$.

4 Backpropagation algorithm for classification

We consider here a K class classification problem. The output of the MLP is $f(x) = \begin{pmatrix} \mathbb{P}(Y = 1/x) \\ . \\ . \\ \mathbb{P}(Y = K/x) \end{pmatrix}$. We assume that the output activation function is the *softmax* function.

$$\text{softmax}(x_1, \ldots, x_K) = \frac{1}{\sum_{k=1}^K e^{x_k}}(e^{x_1}, \ldots, e^{x_K}).$$

Let us make some useful computations to compute the gradient.

$$\begin{aligned} \frac{\partial \text{softmax}(\boldsymbol{x})_i}{\partial x_j} &= \text{softmax}(\boldsymbol{x})_i(1 - \text{softmax}(\boldsymbol{x})_i) \text{ if } i = j \\ &= -\text{softmax}(\boldsymbol{x})_i \text{softmax}(\boldsymbol{x})_j \text{ if } i \neq j \end{aligned}$$

We introduce the notation

$$(f(x))_y = \sum_{k=1}^K \mathbf{1}_{y=k}(f(x))_k,$$

where $(f(x))_k$ is the kth component of $f(x)$: $(f(x))_k = \mathbb{P}(Y = k/x)$. Then we have

$$-\log(f(x))_y = -\sum_{k=1}^K \mathbf{1}_{y=k} \log(f(x))_k = \ell(f(x), y),$$

for the loss function ℓ associated to the cross-entropy.
Using the notations of Section 2.2, we want to compute the gradients

$$\text{Output weights}\frac{\partial \ell(f(x),y)}{\partial W_{i,j}^{(L+1)}} \quad \text{Output biases}\frac{\partial \ell(f(x),y)}{\partial b_i^{(L+1)}}$$
$$\text{Hidden weights}\frac{\partial \ell(f(x),y)}{\partial W_{i,j}^{(h)}} \quad \text{Hidden biases}\frac{\partial \ell(f(x),y)}{\partial b_i^{(h)}}$$

for $1 \leq h \leq L$. We use the chain-rule: if $z(x) = \phi(a_1(x), \ldots, a_J(x))$, then

$$\frac{\partial z}{\partial x_i} = \sum_j \frac{\partial z}{\partial a_j}\frac{\partial a_j}{\partial x_i} = \langle \nabla \phi, \frac{\partial \boldsymbol{a}}{\partial x_i} \rangle.$$

Hence we have

$$\frac{\partial \ell(f(x),y)}{\partial (a^{(L+1)}(x))_i} = \sum_j \frac{\partial \ell(f(x),y)}{\partial f(x)_j}\frac{\partial f(x)_j}{\partial (a^{(L+1)}(x))_i}.$$

$$\frac{\partial \ell(f(x),y)}{\partial f(x)_j} = \frac{-\mathbf{1}_{y=j}}{(f(x))_y}.$$

$$\frac{\partial \ell(f(x),y)}{\partial (a^{(L+1)}(x))_i} = -\sum_j \frac{\mathbf{1}_{y=j}}{(f(x))_y}\frac{\partial \text{softmax}(a^{(L+1)}(x))_j}{\partial (a^{(L+1)}(x))_i}$$
$$= -\frac{1}{(f(x))_y}\frac{\partial \text{softmax}(a^{(L+1)}(x))_y}{\partial (a^{(L+1)}(x))_i}$$
$$= -\frac{1}{(f(x))_y}\text{softmax}(a^{(L+1)}(x))_y(1 - \text{softmax}(a^{(L+1)}(x))_y)\mathbf{1}_{y=i}$$
$$+\frac{1}{(f(x))_y}\text{softmax}(a^{(L+1)}(x))_i\text{softmax}(a^{(L+1)}(x))_y\mathbf{1}_{y\neq i}$$

$$\frac{\partial \ell(f(x),y)}{\partial (a^{(L+1)}(x))_i} = (-1 + f(x)_y)\mathbf{1}_{y=i} + f(x)_i\mathbf{1}_{y\neq i}.$$

Hence we obtain

$$\nabla_{a^{(L+1)}(x)}\ell(f(x),y) = f(x) - e(y),$$

where, for $y \in \{1, 2, \ldots, K\}$, $e(y)$ is the $\mathbb{R}^K$ vector with i th component $\mathbf{1}_{i=y}$.
We now obtain easily the partial derivative of the loss function with respect to the output bias. Since

$$\frac{\partial((a^{(L+1)}(x)))_j}{\partial (b^{(L+1)})_i} = \mathbf{1}_{i=j},$$

$$\nabla_{b^{(L+1)}}\ell(f(x),y) = f(x) - e(y), \tag{10.1}$$

Let us now compute the partial derivative of the loss function with respect to the output weights.

$$\frac{\partial \ell(f(x),y)}{\partial W_{i,j}^{(L+1)}} = \sum_k \frac{\partial \ell(f(x),y)}{\partial (a^{(L+1)}(x))_k}\frac{\partial (a^{(L+1)}(x))_k}{\partial W_{i,j}^{(L+1)}}$$

and

$$\frac{\partial (a^{(L+1)}(x))_k}{\partial W_{i,j}^{(L+1)}} = (h^{(L)}(x))_j\mathbf{1}_{i=k}.$$

Hence

$$\nabla_{W^{(L+1)}}\ell(f(x),y) = (f(x) - e(y))(h^{(L)}(x))'. \tag{10.2}$$

Let us now compute the gradient of the loss function at hidden layers. We use the chain rule

$$\frac{\partial \ell(f(x),y)}{\partial (h^{(k)}(x))_j} = \sum_i \frac{\partial \ell(f(x),y)}{\partial (a^{(k+1)}(x))_i}\frac{\partial (a^{(k+1)}(x))_i}{\partial (h^{(k)}(x))_j}$$

We recall that

$$(a^{(k+1)}(x))_i = b_i^{(k+1)} + \sum_j W_{i,j}^{(k+1)} (h^{(k)}(x))_j.$$

Hence

$$\frac{\partial \ell(f(x),y)}{\partial h^{(k)}(x)_j} = \sum_i \frac{\partial \ell(f(x),y)}{\partial a^{(k+1)}(x)_i} W_{i,j}^{(k+1)}$$

$$\nabla_{h^{(k)}(x)} \ell(f(x),y) = (W^{(k+1)})' \nabla_{a^{(k+1)}(x)} \ell(f(x),y).$$

Recalling that $h^{(k)}(x)_j = \phi(a^{(k)}(x)_j)$,

$$\frac{\partial \ell(f(x),y)}{\partial a^{(k)}(x)_j} = \frac{\partial \ell(f(x),y)}{\partial h^{(k)}(x)_j} \phi'(a^{(k)}(x)_j).$$

Hence,

$$\nabla_{a^{(k)}(x)} \ell(f(x),y) = \nabla_{h^{(k)}(x)} \ell(f(x),y) \odot (\phi'(a^{(k)}(x)_1), \ldots, \phi'(a^{(k)}(x)_j), \ldots)'$$

where $\odot$ denotes the element-wise product. This leads to

$$\begin{aligned} \frac{\partial \ell(f(x),y)}{\partial W_{i,j}^{(k)}} &= \frac{\partial \ell(f(x),y)}{\partial a^{(k)}(x)_i} \frac{\partial a^{(k)}(x)_i}{\partial W_{i,j}^{(k)}} \\ &= \frac{\partial \ell(f(x),y)}{\partial a^{(k)}(x)_i} h_j^{(k-1)}(x) \end{aligned}$$

Finally, the gradient of the loss function with respect to hidden weights is

$$\nabla_{W^{(k)}} \ell(f(x),y) = \nabla_{a^{(k)}(x)} \ell(f(x),y) h^{(k-1)}(x)'. \tag{10.3}$$

The last step is to compute the gradient with respect to the hidden biases. We simply have

$$\frac{\partial \ell(f(x),y)}{\partial b_i^{(k)}} = \frac{\partial \ell(f(x),y)}{\partial a^{(k)}(x)_i}$$

and

$$\nabla_{b^{(k)}} \ell(f(x),y) = \nabla_{a^{(k)}(x)} \ell(f(x),y). \tag{10.4}$$

We can now summarize the backpropagation algorithm.

- **Forward pass**: we fix the value of the current weights $\theta^{(r)} = (W^{(1,r)}, b^{(1,r)}, \ldots, W^{(L+1,r)}, b^{(L+1,r)})$, and we compute the predicted values $f(X_i, \theta^{(r)})$ and all the intermediate values $(a^{(k)}(X_i), h^{(k)}(X_i) = \phi(a^{(k)}(X_i)))_{1 \le k \le L+1}$ that are stored.
- **Backpropagation algorithm**:
  - Compute the output gradient $\nabla_{a^{(L+1)}(x)} \ell(f(x),y) = f(x) - e(y)$.
  - For $k = L+1$ to 1
    - Compute the gradient at the hidden layer k

$$\begin{aligned} \nabla_{W^{(k)}} \ell(f(x),y) &= \nabla_{a^{(k)}(x)} \ell(f(x),y) h^{(k-1)}(x)' \\ \nabla_{b^{(k)}} \ell(f(x),y) &= \nabla_{a^{(k)}(x)} \ell(f(x),y) \end{aligned}$$

    - Compute the gradient at the previous layer

$$\nabla_{h^{(k-1)}(x)} \ell(f(x),y) = (W^{(k)})' \nabla_{a^{(k)}(x)} \ell(f(x),y)$$

and

$$\begin{aligned} \nabla_{a^{(k-1)}(x)} \ell(f(x),y) &= \nabla_{h^{(k-1)}(x)} \ell(f(x),y) \\ &\quad \odot (\ldots, \phi'(a^{(k-1)}(x)_j), \ldots)' \end{aligned}$$

4.1 Optimization algorithms

Many algorithms can be used to minimize the loss function, all of them have hyperparameters, that have to be calibrated, and have an important impact on

the convergence of the algorithms. The elementary tool of all these algorithms is the Stochastic Gradient Descent (SGD) algorithm. It is the most simple one:

$$\theta_i^{new} = \theta_i^{old} - \varepsilon \frac{\partial L}{\partial \theta_i}(\theta_i^{old}),$$

where ε is the *learning rate* , and its calibration is very important for the convergence of the algorithm. If it is too small, the convergence is very slow and the optimization can be blocked on a local minimum. If the learning rate is too large, the network will oscillate around an optimum without stabilizing and converging. A classical way to proceed is to adapt the learning rate during the training: it is recommended to begin with a "large " value of ϵ, (for example 0.1) and to reduce its value during the successive iterations. However, there is no general rule on how to adjust the learning rate, and this is more the experience of the engineer concerning the observation of the evolution of the loss function that will give indications on the way to proceed.

The stochasticity of the SGD algorithm lies in the computation of the gradient. Indeed, we consider *batch learning*: at each step, m training examples are randomly chosen without replacement and the mean of the m corresponding gradients is used to update the parameters. An *epoch* corresponds to a pass through all the learning data, for example if the batch size m is $1/100$ times the sample size n, an epoch corresponds to 100 batches. We iterate the process on a certain number nb of *epochs* that is fixed in advance. If the algorithm did not converge after nb epochs, we have to continue for nb' more epochs. Another stopping rule, called *early stopping* is also used: it consists in considering a validation sample, and stop learning when the loss function for this validation sample stops to decrease. *Batch learning* is used for computational reasons, indeed, as we have seen, the backpropagation algorithm needs to store all the intermediate values computed at the forward step, to compute the gradient during the backward pass, and for big data sets, such as millions of images, this is not feasible, all the more that the deep networks have millions of parameters to calibrate. The batch size m is also a parameter to calibrate. Small batches generally lead to better generalization properties. The particular case of batches of size 1 is called *On-line Gradient Descent*. The disadvantage of this procedure is the very long computation time. Let us summarize the classical SGD algorithm.

Algorithm 8 Stochastic Gradient Descent algorithm

Fix the parameters ε : learning rate, m : batch size, nb : number of epochs.
Choose the initial parameter θ
for $k = 1$ to nb epochs **do**
  for $l = 1$ to n/m **do**
    Take a random batch of size m without replacement in the learning sample: $(X_i, Y_i)_{i \in B_l}$
    Compute the gradients with the backpropagation algorithm

$$g = \frac{1}{m} \sum_{i \in B_l} \nabla_\theta \ell(f(X_i, \theta), Y_i).$$

    Update the parameters : $\theta \leftarrow \theta - \varepsilon g$.
  end for
end for

Since the choice of the learning rate is delicate and very influent on the convergence of the SGD algorithm, variations of the algorithm have been proposed. They are less sensitive to the learning rate. The principle is to add a correction when we update the gradient, called **momentum**. The method is due to Polyak (1964) [31]. The idea is to accumulate an exponentially decaying moving average of past negative gradients and to continue to move in their direction.The momentum algorithm introduces a variable ν, that plays the role of a velocity. An hyperparameter $\alpha \in [0, 1[$ determines how fast the contribution of previous gradients exponentially decay. The method is summarized in Algorithm 9.

Algorithm 9 Stochastic Gradient Descent algorithm with momentum

Fix the parameters ε : learning rate, m: batch size, nb : number of epochs, momentum parameter $\alpha \in [0, 1[$.
Choose the initial parameter θ and the initial velocity ν.
for $k = 1$ to nb epochs **do**
  for $l = 1$ to n/m **do**
    Sample a minibach B of size m from the learning sample.
    Compute the gradient estimate : $g \leftarrow \frac{1}{m} \sum_{i \in B} \nabla_\theta \ell(f(X_i, \theta), Y_i)$.
    Update the velocity : $\nu \leftarrow \alpha\nu - \varepsilon g$.
    Update the parameter : $\theta \leftarrow \theta + \nu$.
  end for
end for

This method allows to attenuate the oscillations of the gradient.
In practice, a more recent version of the momentum due to Nesterov (1983) [30] and Sutskever et al. (2013) [37] is considered, it is called **Nesterov accelerated gradient**. The variants lie in the updates of the parameter and the velocity :

$$\nu \leftarrow \alpha\nu - \varepsilon \frac{1}{m} \sum_{i \in B} \nabla_\theta \ell(f(X_i, \theta + \alpha\nu), Y_i)$$
$$\theta \leftarrow \theta + \nu.$$

The learning rate ε is a difficult parameter to calibrate because it significantly affects the performances of the neural network. This is why new algorithms have been introduced, to be less sensitive to this learning rate : the **RMSProp** algorithm, due to Hinton (2012) [22] and **Adam** (for Adaptive Moments) algorithm, see Kingma and Ba (2014) [24].

The idea of the RMSProp algorithm is to use a different learning rate for each parameter (components of θ) and to automatically adapt this learning rate during the training. It is described in Algorithm 10.

Algorithm 10 RMSProp algorithm

Fix the parameters ε : learning rate, m: batch size, nb : number of epochs, decay rate ρ in $[0, 1[$
Choose the initial parameter θ
Choose a small constant δ, usually 10^{-6} (to avoid division by 0)
Initialize accumulation variable $r = 0$.
for $k = 1$ to nb epochs **do**
  for $l = 1$ to n/m **do**
    Sample a minibach B of size m from the learning sample.
    Compute the gradient estimate : $g \leftarrow \frac{1}{m} \sum_{i \in B} \nabla_\theta \ell(f(X_i, \theta), Y_i)$.
    Accumulate squared gradient $r \leftarrow \rho r + (1 - \rho) g \odot g$
    Update the parameter : $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{\delta + r}} \odot g$ ($\frac{1}{\sqrt{\delta + r}}$ is computed element-wise).
  end for
end for

Adam algorithm (Kingma and Ba, 2014) is also an adaptive learning rate optimization algorithm. "Adam" means "Adaptive moments". It can be viewed as a variant of RMSProp algorithm with momentum. It also includes a bias correction of the first order moment (momentum term) and second order moment. It is described in Algorithm 11.

We have presented the most popular optimization algorithms for deep learning. There is actually no theoretical foundation on the performances of these algorithms, even for convex functions (which is not the case in deep learning problems !). Numerical studies have been performed to compare a large number of optimization algorithms for various learning problems (Schaul et al. (2014)). There is no algorithms that outperforms the other ones. The

Algorithm 11 Adam algorithm

Fix the parameters ε : learning rate, m: batch size, nb : number of epochs, decay rate for moment estimates ρ_1 and ρ_2 in $[0, 1[$
Choose the initial parameter θ
Choose a small constant δ, usually 10^{-6} (to avoid division by 0)
Initialize 1st and 2nd moment variables variable $s = 0,\ r = 0$.
Initial time step $t = 0$.
for $k = 1$ to nb epochs **do**
  for $l = 1$ to n/m **do**
    Sample a minibach B of size m from the learning sample.
    Compute the gradient estimate : $g \leftarrow \frac{1}{m} \sum_{i \in B} \nabla_\theta \ell(f(X_i, \theta), Y_i)$.
    $t \leftarrow t + 1$
    Update first moment estimate $s \leftarrow \rho_1 s + (1 - \rho_1) g$
    Update second moment estimate $r \leftarrow \rho_2 r + (1 - \rho_2) g \odot g$
    Correct biases : $\hat{s} \leftarrow s/(1 - \rho_1^t)$, $\hat{r} \leftarrow r/(1 - \rho_2^t)$
    Update the parameter : $\theta \leftarrow \theta - \frac{\varepsilon}{\sqrt{\hat{r}} + \delta} \odot \hat{s}$
  end for
end for

algorithms with adaptive learning seem more robust to the hyperparameters.

The choice of the initialization of the parameter θ is also an important point. Here are some recommendations : the input data have to be normalized to have approximately the same range. The biases can be initialized to 0. The weights cannot be initialized to 0 since for the $\tanh$ activation function, the derivative at 0 is 0, this is a saddle point. They also cannot be initialized with the same values, otherwise, all the neurons of a hidden layer would have the same behaviour. We generally initialize the weights at random: the values $W_{i,j}^{(k)}$ are i.i.d. Uniform on $[-c, c]$ with possibly $c = \frac{\sqrt{6}}{N_k + N_{k-1}}$ where N_k is the size of the hidden layer k. We also sometimes initialize the weights with a normal distribution $\mathcal{N}(0, 0.01)$ (see Goodfellow et al. (2016) [19]).

5 Regularization

To conclude, let us say a few words about regularization. We have already mentioned $\mathbb{L}^2$ or $\mathbb{L}^1$ penalization; we have also mentioned early stopping. For deep learning, the mostly used method is the **dropout**. It was introduced by Hinton et al. (2012), [22]. With a certain probability p, and independently of the others, each unit of the network is set to 0. The probability p is another hyperparameter. It is classical to set it to 0.5 for units in the hidden layers, and to 0.2 for the entry layer. The computational cost is weak since we just have to set to 0 some weights with probability p. This method improves significantly the generalization properties of deep neural networks and is now the most popular regularization method in this context. The disadvantage is that training is much slower (it needs to increase the number of epochs). Ensembling models (aggregate several models) can also be used. It is also classical to use data augmentation or Adversarial examples. In the course **High Dimensional and Deep Learning** next year, we will see the convolutional neural networks, which are particularly adapted for image classification.

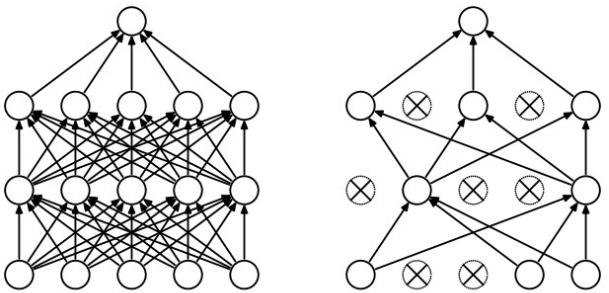


Figure 10.5: Dropout - source: http://blog.christianperone.com/

6 Conclusion

We have presented in this course the feedforward neural networks, and explained how the parameters of these models can be estimated. The choice of the architecture of the network is also a crucial point. Several models can be compared by using a cross validation method to select the "best" model.

The perceptrons are defined for vectors. They are not well adapted for some types of data such as images. By transforming an image into a vector, we loose spatial information, such as forms.

The **convolutional neural networks (CNN)** introduced by Le Cun (1998) [25] have revolutionized image processing. CNN act directly on matrices, or even on tensors for images with three RGB color channels. They are also based on the methods presented in this course to estimate their parameters (backpropagation equations, optimization algorithms ..) The presentation of the convolutional neural networks will studied next year in the High Dimensional and Deep Learning course.

Chapter 11

Imputation of missing data

1 Introduction

Despite the increasing amount of available data and the emergence of the *Big Data*, missing data issues remain a common statistical problem and require a special approach. Ignoring missing data can lead not only to a loss of precision, but also to strong biases in analysis models.

Data are composed of p quantitative or qualitative variables $(Y_1, \ldots, Y_p)$ observed on a sample of n individuals. There are missing data represented by the M matrix called *indication of missing values* [33] whose form depends on the type of missing data.

The main points treated in this chapter are the definition of the different types of missing data and the illustration of their possible distributions, the description of the main strategies for managing missing data by data deletion or completion. The problem is vast and we do not claim to deal with it exhaustively.

2 Types of missing data

In order to properly address the imputation of missing data, it is necessary to distinguish the causes of missing data, especially if they are not simply the result of hazard. A typology has been developed by Little & Rubin (1987), dividing them into 3 categories:

MCAR (*missing completely at random*). A data is MCAR if the probability of absence is the same for all observations. This probability therefore depends only on external parameters independent of this variable. For example: if each participant in a survey decides to answer the income question by rolling a die and refusing to answer if face 6 appears [1]. Note that if the amount of MCAR data is not too large, ignoring observations with missing data will not bias the analysis. However, a loss of precision in the results will probably occur since the model is built with less observations.

MAR (*Missing at random*). The case of MAR data occurs when the data are

not missing completely randomly. Namely, if the probability of missing data is related to one or more other observed variables, it is referred to as "Missing at Random" (MAR). Appropriate statistical methods are available to avoid bias in the analysis (see Section 4).

MNAR (*Missing not at random*) Data is missing not at random (MNAR) if the probability of absence depends on the variable where a missing value occurs. A common example [27] is when people with a large income refuse to give their income. MNAR data induce a loss of precision (inherent to any case of missing data) but also a bias that requires the use of a sensitivity analysis.

2.1 Distribution of missing data

Let $Y = (y_{ij}) \in \mathbb{R}^{n \times p}$ be the rectangular matrix of data for p variables $Y_1, \ldots, Y_p$ and n observations. Let us consider $M = (m_{ij})$ the matrix of indication of the missing values [33], which will define the distribution of the missing data. We will then consider 3 types of distribution :

1. Univariate missing values. Missing values occur only for one variable Y_k. If an observation y_{ik} is missing, then there will be no more observations of this variable, the observations y_{lk} for $l \geq i$ are missing. An illustration is given in Figure 11.1 (case a)).
2. Missing values are said to be **monotones** if Y_j missing for an individual i implies that all the following variables $\{Y_k\}_{k>j}$ are missing for this individual (Figure 11.1, case b)). The missing data indicator M is then an integer $M \in \{1, 2, \ldots, p\}$ for each individual, indicating the largest j for which Y_j is observed.
3. The missing values are **not monotonic** (or **arbitrary**), as shown in Figure 11.1, case c). In this case, the matrix of missing values is defined by $M = (m_{ij})$ with $m_{ij} = 1$ if y_{ij} is missing and zero otherwise.

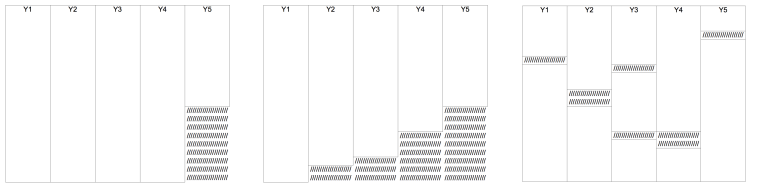


Figure 11.1: Distribution of missing data. (a) univariate, (b) monotonous, (c) arbitrary/non-monotonous

In the case of longitudinal data (see figure 11.2), the monotonic distribution corresponds to a right censoring.

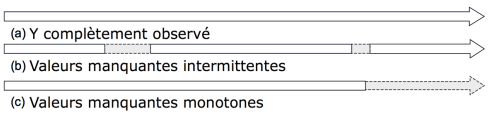


Figure 11.2: Distributions of missing data for longitudinal variables. (a) full set, (b) arbitrary/non-monotonic and (c) monotonic

2.2 Probability of absence

The probability of absence according to the type of missing data (MCAR, MAR, MNAR) can be expressed in terms of the matrix M [33]. The data are divided in two according to the M matrix of missing data. Therefore, $Y_{obs} = Y\mathbf{1}_{\{M=0\}}$ is defined as observed data and $Y_{mis} = Y\mathbf{1}_{\{M=1\}}$ as missing data. Finally $Y = \{Y_{obs}, Y_{mis}\}$. The missing data mechanism is characterized by the conditional distribution $p(M|Y)$.

- In the case of **MCAR** data, the absence of data does not depend on Y

values so

$$p(M|Y) = p(M).$$

- In the case of **MAR**, the absence of data depends only on Y_{obs} :

$$p(M|Y) = p(M|Y_{obs}) \text{ for all } Y_{mis}.$$

- Finally, the data are **MNAR** if the distribution of M also depends on Y_{mis}.

Example for a univariate sample

Let $Y = (y_1, \ldots, y_n)^\top$ where y_i is the observation of a random variable for the individual i, and $M = (M_1, \ldots, M_n)$ where $M_i = 0$ for observed data and $M_i = 1$ for missing data. It is also assumed that the joint distribution is independent of the individuals. So

$$p(Y, M) = p(Y)p(M|Y) = \prod_{i=1}^{n} p(y_i) \prod_{i=1}^{n} p(M_i|y_i)$$

where $p(y_i)$ is the density of y_i and $p(M_i|y_i)$ is the density of a Bernoulli distribution with the probability $\mathbb{P}(M_i = 1|y_i)$ that y_i is missing.

If $\mathbb{P}(M_i = 1|y_i) = \alpha$ with α a constant that does not depend on y_i then it is a MCAR (or in this case also MAR) case. If $\mathbb{P}(M_i = 1|y_i)$ depends on y_i then the missing data mechanism is MNAR.

3 Analysis without completion

3.1 Methods with data deletion

In some cases, analysis is possible without imputing missing data. In general, two classical methods are used :

- **Analysis of complete cases**, which consists in considering only those individuals for whom all the data are available, i.e. by deleting lines with missing values. This is done automatically with R (`na.action=na.omit`). This method, as can be seen in 11.3, may delete too much data and hence greatly increase the loss of precision. In addition, if the data are not MCAR, removing observations will bias the analysis since the subsample of cases represented by the complete data may not be representative of the original sample.

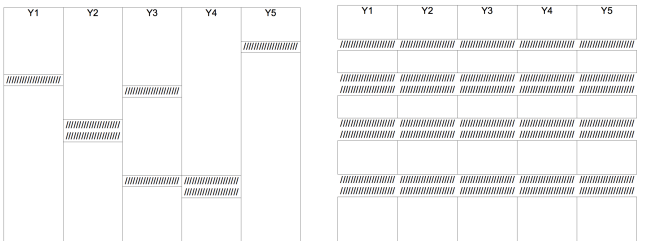


Figure 11.3: Distribution of missing data. (a) original data with arbitrary missing values, (b) remaining observations in complete case analysis

- **Analysis of available cases**. In order to avoid deleting too much data, it is possible to do *available-case analysis*. Different aspects of the problem are then studied with different sub-samples. However, the different analyses will not necessarily be compatible with each other. For example, if a variable Y_1 is available for all individuals and a variable Y_2 only for 80% of the individuals, we may estimate the distribution of Y_1 with all the individuals and the distribution of Y_2 with 80% of the individuals. The available-case analysis also refers to the case where a variable is removed from the dataset because it has too many missing values.

3.2 Methods tolerant to missing data

While most methods automatically remove missing data, some tolerate them. This is the case for example of trees (CART) which consider *surrogate splits* : At each node splitting, several optimal pairs variable/threshold are considered and memorized. To compute a prediction, if the data is missing for an observation, it is not the best division that is used but the one just after.

4 Imputation Methods

This section provides a non-exhaustive overview of the most common completion methods. A dataset consists of p quantitative or qualitative variables $(Y_1, \ldots, Y_p)$ observed for a sample of n individuals; M refers to the matrix indicating missing values by $m_{ij} = \mathbf{1}_{\{y_{ij}.\text{missing}\}}$

4.1 Stationary completion

There are several possible stationary completions : the most frequently represented value (*Concept Most Common Attribute Value Fitting*, CMCF [41]) or simply the last known value (*Last observation carried forward*, LOCF). This method may seem too naive but is often used to lay the foundation for a comparison between completion methods.

4.2 Completion by a linear combination of observations

Another common technique is to replace all missing values with a linear combination of observations. Let us mention the imputation by the mean : the missing value Y_{ij} is replaced by the mean $\bar{Y}_j$ over all observed values of the variable Y_j. This case is generalized to any weighted linear combination of the observations. The median of Y_j can also be considered.

Instead of using all available values, it is possible to restrict oneself to methods that select the most influential values by local aggregation or regression or even by combining different aspects.

4.3 Nearest Neighbor Method (KNN)

The completion by *k-nearest neighbors* or KNN consists in running the following algorithm that models and predicts the missing data. Assume that the values $Y_{i^\star,J}$ are missing, where J is the subset of variables not observed for the individual $i^\star$.

Algorithm 12 Algorithm of k-nearest neighbors (k-nn)

Choice of an integer $1 \leq k \leq n$.
Computation of the distances $d(Y_{i^\star}, Y_i)\,, \quad i = 1, \ldots, n$ (using only the observed variables for $Y_{i^\star}$ to compute the distances).
Retrieve the k observations $Y_{(i_1)}, \ldots, Y_{(i_k)}$ for which these distances are the smallest.
Assign to the missing values the average of the values of the k-nearest neighbors :

$$\forall j^\star \in J, Y_{i^\star j^\star} = \frac{1}{k}\left(Y_{(i_1),j^\star} + \ldots + Y_{(i_k),j^\star}\right)$$

The nearest neighbors method requires the choice of the parameter k by optimization of a criterion. Moreover, the notion of distance between individuals must be chosen carefully. One generally considers the Euclidean or Mahalanobis distance.

4.4 Local regression

The LOcal regrESSion : LOESS [42] also allows to impute missing data. For this, a polynomial with small degree is fitted around the missing data by weighted least squares, giving more weight to values close to the missing data.

Let $Y_{i^\star}$ be an observation with q (among p) missing values. These missing values are imputed by local regression following the algorithm below :

Algorithm 13 Algorithm LOESS

Getting the k nearest neighbors $Y_{(i_1)}, \ldots, Y_{(i_k)}$.
Creation of matrices $A \in \mathbb{R}^{k\times(p-q)}$, $B \in \mathbb{R}^{k\times q}$ and $w \in \mathbb{R}^{(p-q)\times 1}$ such that :

- Lines of A correspond to the k nearest neighbors deprived of the values at the indices of the missing variables for $Y_{i^\star}$.
- The columns of B correspond to the values of the neighbors for the indices of the missing variables for $Y_{i^\star}$.
- The vector $w = (Y_{i^\star})_{obs}$ corresponds to the $(p-q)$ observed values of $Y_{i^\star}$.

Solving the Least Square Problem

$$x^\star = \text{argmin}_{x\in\mathbb{R}^k} \parallel A^\top x - w \parallel$$

where $\parallel \cdot \parallel$ is the quadratic standard of $\mathbb{R}^k$.
The vector of missing data is then predicted by

$$(Y_{i^\star})_{mis} = B^\top x^\star.$$

4.5 By singular value decomposition (SVD)

Cases where there is sufficiently observed data

If there are much more observed data than missing ones, the dataset Y is separated into two groups : on one side Y^c with the complete observations and on the other side Y^m including the individuals for which some data are missing. We then consider the truncated singular value decomposition (SVD) of the complete set Y^c (see [15]) :

$$\hat{Y_J}^c = U_J D_J V_J^\top$$

where D_J is the diagonal matrix including the J first singular values of Y^c. Note that $V_J \in \mathcal{M}_{p,J}(\mathbb{R})$. The missing values are then imputed by regression. More precisely, let $(Y_{i^\star})_{obs} \in \mathbb{R}^{p-q}$ the set of observed values for $Y_{i^\star}$, let $V_J^\star$ be the truncated version of V_J, i.e. for which the lines corresponding to the q missing variables for $Y_{i^\star}$ have been deleted. Hence $V_J^\star \in \mathcal{M}_{p-q,J}(\mathbb{R})$. Then the prediction of the q missing data for the individual $i^\star$, $(Y_{i^\star})_{mis}$ is given by

$$(Y_{i^\star})_{mis} = V_J^{(\star)}\hat{\beta},$$

where $V_J^{(\star)} \in \mathcal{M}_{q,J}(\mathbb{R})$ is the complement of $V_J^\star$ in V_J and

$$\hat{\beta} = (V_J^{\star\top} V_J^\star)^{-1} V_J^{\star\top} (Y_{i^\star})_{obs}.$$

As for KNN, this method requires the choice of the parameter J.

Cases with too many missing data

If there are too many missing data, this will induce a significant bias in the calculation of the SVD decomposition. In addition, there may be at least one missing data for all observations. In this case, the following problem must be solved :

$$\min_{U_J, V_J, D_J} \parallel Y - m - U_J D_J V_J^\top \parallel_\star \tag{11.1}$$

where $\parallel \cdot \parallel_\star$ sums the squares of the elements of the matrix, ignoring the missing values and m is the vector of the means of the observations. The resolution of this problem follows the following algorithm :

Algorithm 14 Algorithm of Completion by SVD

Create a matrix Y^0 for which the missing values are completed by the mean.
Calculate the SVD solution of the problem (11.1) for the completed matrix Y^k. We thus create Y^{k+1} by replacing the missing values of Y by those of the regression.
Iterate the previous step until $\| Y^k - Y^{k+1} \| / \| Y^k \| < \epsilon$, arbitrary threshold (often at 10^{-6}).

4.6 Use of Random Forests

Stekhoven and Bühlmann (2011)[12] have proposed a completion method based on random forests called `missForest`. An R library of the same name is associated with it. This method requires a first naive imputation, by default a completion by the mean, in order to obtain a complete learning sample. Then a series of random forests are adjusted until the first degradation of the model.

To formalize this, the initial dataset is separated into four parts. For each variable Y^s, $s = 1, \ldots, p$ whose missing values are indexed by $i^s_{mis} \subseteq \{1, \ldots, n\}$, one defines

1. y^s_{obs} the observed values for Y^s.
2. y^s_{mis} the missing values for Y^s.
3. $X^s = Y \setminus Y^s$ the set of regressors of Y^s among which we consider
   (a) x^s_{obs} the observed regressors for $i^s_{obs} = \{i, \ldots, n\} \setminus i^s_{mis}$
   (b) x^s_{mis} all the regressors (observed and missing) for i^s_{mis}

The method then follows the following algorithm :

MissForest Algorithm

1. First "naive" completion of missing values.
2. Let $\mathcal{K}$ be the vector of column indices of Y sorted by increasing amount of missing values;
3. **while** γ is not reached **do**
   (a) Y^{old}_{imp} = previously imputed matrix
   (b) **for** s in $\mathcal{K}$ **do**
      i. Adjust $y^{(s)}_{obs} \sim x^{(s)}_{obs}$ by a random forest
      ii. Predict $y^{(s)}_{mis}$ with the regressors $x^{(s)}_{mis}$.
      iii. Y^{new}_{imp} is the new matrix completed by the predicted values $y^{(s)}_{mis}$
   (c) **end for**
   (d) update the γ criterion
   (e) **end while**
4. **return** the imputed matrix Y_{imp}.

The stopping criterion γ is reached as soon as the difference between the newly imputed matrix and the previous one increases for the first time. The difference of the set of continuous variables N is defined as

$$\Delta_N = \frac{\sum_{j \in N} \left(Y^{new}_{imp} - Y^{old}_{imp} \right)^2}{\sum_{j \in N} \left(Y^{new}_{imp} \right)^2}$$

For the set of qualitative variables F, the difference is defined by

$$\Delta_F = \frac{\sum_{j \in F} \sum_{i=1}^{n} \mathbf{1}_{Y_{imp}^{new} \neq Y_{imp}^{old}}}{\#NA}$$

where $\#NA$ is the number of missing values in the categorical variables.

4.7 Bayesian Inference

Let θ be the realization of a random variable and let $p(\theta)$ be its distribution *a priori*. The distribution *a posteriori* is thus given by:

$$p(\theta|Y_{obs}) \propto p(\theta) f(Y_{obs}/\theta)$$

Tanner and Wong's (1987) *data augmentation* method iteratively simulates random samples of missing values and model parameters, taking into account the observed data at each iteration, consisting of an imputation step (I) and a "posterior" step (P).

Let $\theta^{(0)}$ be an initial draw obtained from an approximation of the posterior distribution of θ. For a value of $\theta^{(t)}$ of θ at an instant t

- **Imputation (I)** : simulate $Y_{mis}^{(t)}$ with a density $p(Y_{mis}|Y_{obs}, \theta^{(t)})$.
- **Posterior (P)** : simulate $\theta^{(t+1)}$ with a density $p(\theta|Y_{obs}, Y_{mis}^{(t)})$

This iterative procedure converges to a draw of the joint distribution of $(Y_{mis}, \theta|Y_{obs})$ when $t \to +\infty$.

4.8 Multiple Imputation

Multiple imputation consists, as its name suggests, of imputing missing values several times in order to combine the results to reduce the error due to the imputation [11]. It also allows to define a measure of the uncertainty induced by the completion.

Maintaining the original variability of the data is done by creating imputed values that are based on variables correlated with missing data and causes of absence. Uncertainty is taken into account by creating different versions of missing data and observing the variability between imputed data sets.

Amelia II

Amelia II is a multiple imputation program for continuous variables developed by James Honaker et al (2011) [23]. The model is based on an assumption of normality : $Y \sim \mathcal{N}_k(\mu, \Sigma)$, and thus sometimes requires prior transformations of the data.

Let M be the matrix indicating the missing data and $\theta = (\mu, \Sigma)$ the parameters of the model. Another hypothesis is that the data are **MAR** so

$$p(M|Y) = p(M|Y_{obs})$$

The likelihood $p(Y_{obs}|\theta)$ is then written as follows

$$p(Y_{obs}, M|\theta) = p(M|Y_{obs}) p(Y_{obs}|\theta)$$

So

$$p(\theta|Y_{obs}) \propto p(Y_{obs}|\theta)$$

Using the iterative property of the expectation,

$$p(Y_{obs}|\theta) = \int p(Y|\theta) dY_{mis}$$

We thus obtain the a posteriori distribution

$$p(\theta|Y_{obs}) \propto p(Y_{obs}|\theta) = \int p(Y|\theta) dY_{mis}$$

Amelia II's EMB algorithm combines the classical EM algorithm (for maximum likelihood) with a bootstrap approach. For each run, the data are estimated by *bootstrap* to simulate the uncertainty then the EM algorithm is run to find the *a posteriori* estimator $\hat{\theta}_{MAP}$ for the *bootstrap* data. Imputations are then created by drawing Y_{mis} according to its conditional distribution on Y_{obs} and simulations of θ.

5 Examples

5.1 Gas consumption frauds

The different completion methods were tested and compared on an example of gas consumption fraud detection. Let $Y \in \mathbb{R}^{n\times 12}$ such that y_{ij} is the individual's gas consumption for individual i in month j. The distribution of the missing data is non-monotonic and we assume MAR data. After a log transformation in order to approach normality, completion was performed. The results were compared with a test sample of 10% of the data, previously removed from the set.

This actual data set had at least one missing value per individual, and a total of 50.4% of the data was missing. If we consider only the individual monthly consumption, we obtain the error distribution of each method shown in Figure 11.4.

5.2 Parisian stock market outstanding (EBP)

We are interested in the prices of stock market assets on the Paris market from 2000 to 2009. We consider 252 prices of companies or indexes regularly quoted over this period. By limiting ourselves to the MCAR case, we artificially create more and more missing data to impute. For 10% of missing data, a comparison of imputation methods is given Figure 11.5. Three methods outperform the others : SVD, missForest and Amelia II.

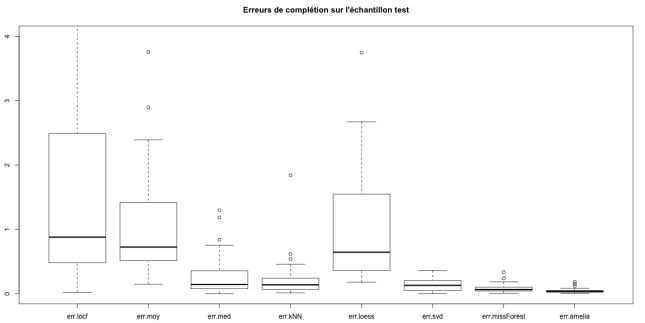


Figure 11.4: Fraud - Completion errors on a test sample

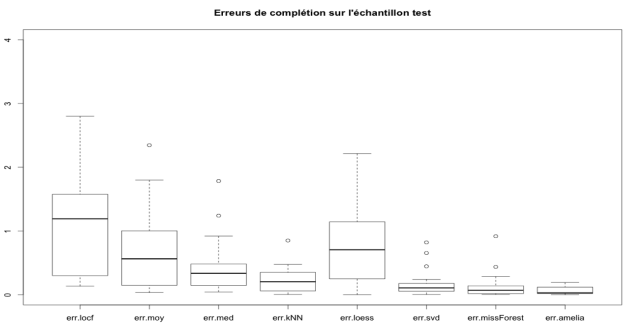


Figure 11.5: EBP - Completion errors on a 10% test sample

The robustness of these methods was tested by gradually increasing the amount of missing data. The results are given Figure 11.6 for Amelia II and Figure 11.7 for missForest.

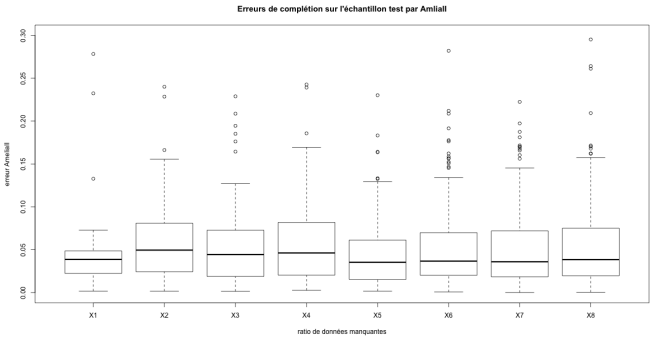


Figure 11.6: EBP - Completion errors on a test sample by Amelia II when the amount of missing values increases

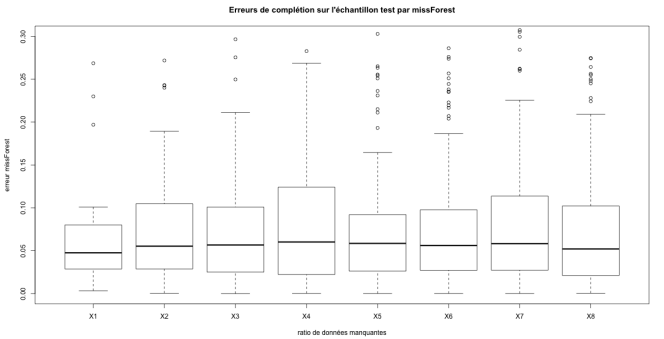


Figure 11.7: EBP - Completion errors on a test sample by missForest when the amount of missing values increases

5.3 Coronary Heart Disease (CHD)

Most imputation methods are defined only for quantitative variables. However, some of the methods presented above can be used to impute qualitative or even heterogeneous data. This is the case of LOCF, KNN and missForest, which were therefore tested on a reference data set on completion problems [29]. The data were acquired by Detranao et al (1989) [32] and made available by Bache and Lichman (2013)[4]. They are in the form of a matrix of medical observations $Y \in \mathbb{R}^{n\times 14}$ of 14 heterogeneous variables for n patients. The dataset thus contains quantitative (`age`, `pressure`, `cholesterol`, `maximum heart rate`, `oldpeak`) and qualitative variables (`sex`, `pain`, `sugar`, `cardio`, `angina`, `peak slope`, `number of heart vessels`, `thalassemia`, `absence/presence of heart disease`).

By always limiting oneself to the MCAR case, one artificially creates more and more missing data to be imputed. The adequacy of the imputation is given by the mean of the error in absolute value in the case of quantitative data and by the Hamming distance in the case of qualitative data. The results are shown in Figure 11.8.

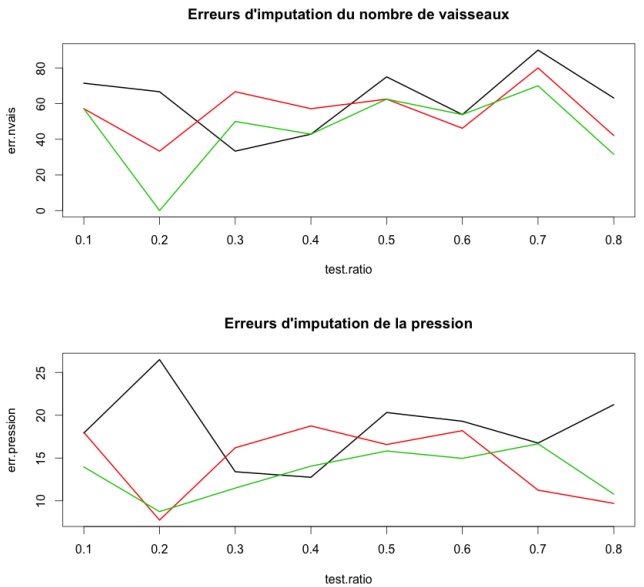


Figure 11.8: CHD - Completion errors on a test sample by LOCF (black), KNN (red) and missForest (green) when the amount of missing values increases, for a qualitative (above) and quantitative (below) variable

.

Chapter 12

Anticiper les Risques Juridiques des Systèmes d'IA

Résumé

Suite à la publication du livre blanc pour une approche de l'IA basée sur l'excellence et la confiance, la Commission Européenne (CE) a publié de nombreuses propositions de textes réglementaires dont un (AI Act) (CE 2021) établissant des règles harmonisées sur l'intelligence artificielle *(IA). Quels seront les conséquences et impacts de l'adoption à venir de ce texte du point de vue d'un mathématicien ou plutôt statisticien impliqué dans la conception de système d'intelligence artificielle (IA) à haut risque au sens de la CE? Quels outils et méthodes permettent de répondre aux obligations à venir de conformité: analyse rigoureuse et documentée des données traitées, des performances, robustesse, résilience de l'algorithme, de son explicabilité, des risques, pour les droits fondamentaux, de biais discriminatoires? Ces questions sont illustrées par un exemple numérique analogue à un score de crédit (cf. tutoriel) à la recherche d'un moins mauvais compromis entre toutes les contraintes. Nous concluons sur les avancées et limites du projet de règlement pour les systèmes d'IA à haut risque.*

1 Introduction

L'adoption en 2018 du Règlement Général de la Protection des Données (RGPD) a profondément modifié les comportements et pratiques des entreprises dans leurs gestions des données, messageries et sites internet. Néanmoins, les condamnations récurrentes des principaux acteurs du numérique, notamment pour abus de position dominante, apportent les preuves de l'inutilité des chartes (*softlaw*) et résolutions éthiques (*ethical washing*). En conséquence et résistant aux accusations fallacieuses de freiner la recherche, l'Europe poursuit sa démarche visant à harmoniser réglementations et innovations technologiques pour le respect des droits humains fondamentaux mais aussi la défense des intérêts commerciaux de l'Union.

La publication par la Commission Européenne (CE) d'un livre blanc sur l'Intelligence Artificielle: une approche européenne axée sur l'excellence et la confiance (CE 2020) fait suite au guide pour une IA digne de confiance

rédigé par un groupe d'experts (CE 2019). L'étape suivante est la publication de propositions de règlements dont certains en cours d'adoption:

- *Digital Market Act* (2020): recherche d'équité dans les relations commerciales et risques d'entraves à la concurrence à l'encontre des entreprises européennes;
- *Digital Services Act* (2020): sites de service intermédiaire, d'hébergement, de plateforme en ligne et autres réseaux sociaux; comment contrôler les contenus illicites et risques des outils automatiques de modération;
- *Data Governance Act* (2020) contractualisation des utilisations, réutilisations, des bases de données tant publiques que privées (fiducie des données);
- *Artificial Intelligence Act* (CE 2021): proposition de règlement établissant des règles harmonisées sur l'intelligence artificielle.

S'ajoutant au RGPD pour la protection des données à caractère personnel, l'adoption européenne à venir de ce dernier texte (*AI Act*) va profondément impacter les conditions de développements et d'exploitations des systèmes d'Intelligence Artificielle (systèmes d'IA). Cette démarche fait passer d'une IA souhaitée éthique (*ethical AI*), à une *obligation de conformité* (*lawfull AI*) qui confère le marquage "CE" ouvrant l'accès au marché européen. La CE veut ainsi manifester son *leadership* normatif à l'international afin que ce pouvoir de l'UE sur la réglementation et le marché lui confère un avantage concurrentiel dans le domaine de l'IA.

En conséquence, le présent document propose une réflexion sur la prise en compte méthodologique de ce projet de réglementation concernant plus spécifiquement les compétences usuelles en Statistique, Mathématiques, des équipes de développement d'un système d'IA, notamment ceux jugés à haut risque selon les critères européens. Il cible plus particulièrement certaines des sept exigences citées dans le guide des experts (CE 2019), reprises dans le livre blanc (CE 2020) et identifiées comme risques potentiels (Besse et al. 2019): 1. confidentialité et analyse des données ; 2. précision, robustesse, résilience ; 3. explicabilité ; 4. non-discrimination.

La section 2 suivante extrait de l'*AI Act* les éléments clefs impactant les choix et développements méthodologiques puis la section 3 en commente les conséquences tout en proposant les outils statistiques bien connus de niveau Master et bagage d'un futur *scientifique des données*. Ceux ci semblent adaptés voire suffisants pour satisfaire aux futures obligations réglementaires de contrôle des risques afférents aux systèmes d'IA. Enfin la section 4 déroule un cas d'usage numérique analogue à la prévision d'un score de crédit sur un jeu de données concret. Cet exemple, extrait d'un tutoriel dont le code est librement accessible, permet d'illustrer la démarche de recherche d'un moins mauvais compromis à élaborer entre confidentialité, performance, explicabilité et sources de discrimination. Il souligne les difficultés soulevées par la rédaction de la documentation qui devra accompagner tout système d'IA à haut risque. En conclusion, nous proposons une synthèse des principales avancées de ce projet d'*AI Act* et en relevons, dans la version d'avril 2021, les principales limites.

2 Impacts techniques de l'*AI Act*

2.1 Structure du projet de règlement

Castets-Renard et Besse (2022) détaillent une analyse du régime de responsabilité *ex ante*[1] proposé dans les 89 considérants[2] et 85 articles structurés en 12 titres de l'*AI Act*: entre auto-régulation, certification, normalisation, pour définir des règles de conformité notamment pour la défense des droits fondamentaux. L'objectif du présent article est plus spécifique, il est focalisé sur les éléments du projet de réglementation concernant directement le statisticien ou scientifique des données impliqué dans la conception d'un système d'IA jugé à haut risque car impactant des personnes physiques.

De façon générale, les considérants, introductifs au projet, listent donc les principes retenus par la CE et qui ont prévalu à la rédaction des articles. La CE insiste sur la nécessité de la construction de normes internationales en priorisant le respect des droits fondamentaux dont la non-discrimination. Consciente de la place occupée par les algorithmes d'apprentissage statistiques, elle souligne la nécessité de la représentativité statistique des données d'entraînement et l'importance d'une documentation exhaustive à propos de ces données et des performances d'un système d'IA. Consciente également de l'opacité de ces algorithmes, elle demande que les capacités d'interprétation de leurs sorties ou décisions en découlant soient à jour des recherches scientifiques en cours et qu'un suivi puisse être assuré grâce à une journalisation ou archivage des décisions et données afférentes.

2.2 Articles les plus concernés

La définition adoptée de l'IA (art. 3) est pragmatique et très flexible en se basant sur la liste exhaustive des algorithmes concernés (annexe I). Les algorithmes d'apprentissage automatique supervisés ou non, par renforcement, constituent actuellement l'essentiel des applications quotidiennes de l'IA. La représentation de connaissances, la programmation inductive et plus généralement les systèmes experts très développés dans les années 70s, restent présents dans certains domaines. Le troisième type d'algorithme cité cible les approches statistiques, inférences bayésiennes et méthodes d'optimisation. Les approches statistiques bayésiennes ou non conduisant très généralement à des prévisions pour l'aide à la décision peuvent être incluses dans la grande famille de l'apprentissage fondée sur des données. En revanche, les méthodes d'optimisation comme par exemple celles d'allocation optimale de ressources des sites d'intermédiation (*e.g.* Uber, ParcourSup,...) nécessitent une approche particulière. Cette liste peut être facilement adaptée en fonction des évolutions technologiques. Ces définitions reconnaissent la place prépondérante de l'apprentissage statistique et donc des données exploitées pour leur construction. Ils laissent de côté les algorithmes procéduraux basés sur les règles logiques d'une législation comme par exemple ceux présidant aux calculs des montants d'allocations.

Les articles 5 et 6 adoptent également le principe de définitions pragmatiques en listant explicitement les applications prohibées (art. 5) et celles à haut risque de l'IA facilement adaptables en fonction des évolutions technologiques. L'article 6 fait la différence entre les systèmes faisant déjà l'objet d'une réglementation européenne (annexe II: systèmes de transports et de soins) qui nécessitent une certification *ex-ante* par un tiers, organisme de notification, contrairement aux autres (annexe III) impactant également des personnes physiques mais dont le processus de mise en conformité est seulement déclaratif. *Attention*: la consultation attentive de ces annexes, de leur évolution, est importante pour bien distinguer les systèmes à haut risque des autres. Les scores de crédit bancaire sont concernés (cf. exemple numérique section 4) ainsi que les évaluations *individuelles* de "police prédictive" ou les scores de récidive (justice) mais pas explicitement celles concernant des évaluations

[1] Par opposition à *ex-post*, *ex-ante* signifie ici que l'analyse ou audit de conformité d'un algorithme d'IA afin de valider sa certification (marquage "CE") est considérée ou effectivement réalisée *avant* sa diffusion ou commercialisation et donc avant sa mise en exploitation.

[2] Les considérants sont une liste de principes qui motivent un décret, une loi ou un règlement et qui en précèdent le texte contenu dans la liste des articles.

de risques de délits par bloc géographique telles *Predpol* ou Paved en France. Pour les applications dans le domaine de la justice, seuls sont concernés les systèmes d'IA à l'usage des autorités judiciaires (magistrats) tels le projet abandonné DataJust mais pas ceux à l'usage des cabinets d'avocats (*e.g. case law analytics*).

L'article 10 est fondamental, il insiste sur l'importance d'une exploration statistique préalable exhaustive des données avant de lancer les procédures largement automatiques d'apprentissage et optimisation. Il évite une forme d'hypocrisie en autorisant, sous réserve de précautions avancées pour la confidentialité, la constitution de bases de données personnelles sensibles permettant par exemple des statistiques ethniques. Cela autorise la mesure directe des biais statistiques, sources potentielles de discrimination.

L'article 11 impose la rédaction d'une documentation qui est essentielle pour ouvrir la possibilité d'audit *ex-ante* d'un système d'IA à haut risque relevant de l'annexe II ou celui d'un contrôle *ex-post* pour ceux relevant de l'annexe III. Avec un reversement de la charge de preuve, c'est au concepteur de montrer qu'il a mis en œuvre ce qu'il était techniquement possible en matière de sécurité, qualité, explicabilité, non discrimination, pour atteindre les objectifs attendus de conformité.

L'article 12 impose un archivage ou journalisation du fonctionnement d'un système d'IA à haut risque. Cette obligation est nouvelle par rapport aux textes européens précédents. Elle est indispensable pour assurer le suivi des mesures de performances, de risques et donc pour être capable de détecter des failles nécessitant des mises à jour voire un ré-entraînement du système ou même son arrêt. Les conditions d'archivage sont précisées dans l'article 61 (*post-market monitoring*).

Selon l'article 13 un utilisateur devrait pouvoir interpréter les sorties, et doit être clairement informé des performances, éventuellement en fonction des groupes concernés, ainsi que des risques notamment de biais et donc de discrimination. Il s'agit ici d'un point sensible directement dépendant de la complexité des systèmes d'IA à base d'algorithmes sophistiqués donc opaques d'apprentissage statistique. Le choix des métriques de biais sont laissées à l'initiative du concepteur. De plus, le manque de recul sur les recherches en cours en matière d'explicabilité d'une décision algorithmique laissent beaucoup de latitude à l'interprétation de cet article qui devra être adaptée à l'évolution des recherches très actives sur ce thème. L'article 14 complète ces dispositions en imposant une surveillance humaine visant à prévenir ou minimiser les risques pour la santé, la sécurité ou les droits fondamentaux.

L'article 15 comble une lacune importante par l'obligation de déclaration des performances (précisions, robustesse, résilience) d'un système d'IA à haut risque. Il concerne également les algorithmes d'apprentissage par renforcement soumis à des risques spécifiques: dérives potentielles (biais) et attaques malveillantes (cybersécurité) comme ce fut le cas pour le *chatbot Tay* de *Microsoft*.

Les articles des chapitres suivants du Titre III notifient des obligations sans apporter de précisions techniques ou méthodologiques: obligations faites au fournisseur (art. 16), obligation de mise en place d'un système de gestion de la qualité (art. 17), notamment de toute la procédure de gestion des données de la collecte initiale à leurs mises à jour en exploitation, ainsi que de la maintenance post-commercialisation; obligation de documentation technique (art. 18), d'évaluation de la conformité (art. 19), obligation des utilisateurs (art. 29)...

Les États membres sont, par ailleurs, invités à désigner une autorité notifiante comme responsable du suivi des procédures relatives aux systèmes à haut risque et un organisme notifié (art. 30 à 39) indépendant, tout à fait classique des mécanismes de certification déjà en œuvre. Un marquage "CE" sera délivré aux systèmes conformes (art. 49).

Ce processus de marquage "CE" est essentiel pour les systèmes d'IA à haut

risque de l'annexe II, il repose sur un audit *ex-ante* requérant, dans le cas d'une évaluation externe, des compétences très élaborées de la part de l'organisme qui en porte la responsabilité afin d'être à même de pouvoir déceler des manquements intentionnels ou non. Sans évaluation externe, pour les systèmes d'IA de l'annexe II, c'est à l'utilisateur de prendre ses responsabilités vis-à-vis du respect, entre autres, des droits fondamentaux afin de pouvoir faire face à un contrôle si l'État membre désigne une autorité compétente à ce sujet et lui en fournit les moyens.

2.3 Conséquences

L'analyse de ces quelques articles amène des commentaires ou questions, notamment sous le prisme d'une approche mathématique ou statistique de conception d'un système d'IA.

Projet Le projet de règlement (`AI Act`) entre dans un long processus (3 ou 4 ans comme le RGPD?) de maturation avant une adoption européenne et une application par les États membres. Les amendements à venir devront être successivement pris en considération pour en analyser les conséquences en espérant que des réponses, précisions, corrections, seront apportées aux points ci-dessous. Néanmoins et compte tenu des temps et coûts de conception d'un système d'IA, il est important d'anticiper dès maintenant l'adoption de ce cadre réglementaire.

Exigences essentielles À la suite du guide des experts, le livre blanc appelle à satisfaire sept *exigences essentielles* dont celles de non discrimination et équité, bien être sociétal et environnemental.

Environnement la prise en compte de l'impact environnemental reste anecdotique, simplement évoquées dans les considérants (28) et (81), puis l'article 69 (*codes de conduite*) 2. sans aucune obligation formelle de calculer une balance bénéfices / risques (environnementaux ou autres) d'un système d'IA. Ainsi, l'obligation de l'archivage des données de fonctionnement d'un système d'IA génère un coût environnemental qui mériterait d'être pris en compte dans les risques afférents à son déploiement au regard de son utilité.

Équité La demande exprimée qu'un système d'IA satisfasse au respect des droits fondamentaux en référence à la charte de l'UE, notamment celui de non-discrimination, est très présente dans le livre blanc (cité 16 fois), comme dans les considérants (15, 17, 28, 39) de la proposition de règlement. En revanche, ce principe n'apparaît plus explicitement dans les articles. Est-ce sa présence dans des textes de plus haut niveau comme la Charte des Droits Fondamentaux de l'UE qui n'a pas justifié ici une répétition ou encore un manque d'harmonisation entre les États membres à ce propos? Il n'y a donc pas de précision sur les façon de "mesurer" une discrimination ou la nécessité de l'atténuer. En revanche, les recherches et documentations des biais potentiels sont clairement explicitées.

Normes Le considérant (13) appelle à la définition de normes internationales notamment à propos des droits fondamentaux. En l'absence d'une définition juridique de l'équité d'un algorithme, celle-ci est définie en creux par l'*absence de discrimination* interdite explicitement. Le souci est que la littérature regorge de dizaines de définitions de biais statistiques pouvant être à l'origine de sources de discrimination; lesquels considérer en priorité? Il est peu probable que les autorités compétentes se prononcent à ce sujet, elles se focalisent (LNE 2021) sur les mesures de performances des systèmes d'IA de l'annexe II, notamment les systèmes de transport et les dispositifs de santé en vue de leur certification (marquage "CE").

Néanmoins, la recherche d'un biais systémique ou de société est requise dans l'analyse préalable des données (art. 10, 2. (f)), ainsi que l'obligation de détailler les performances (précision) par groupe ou sous groupe d'un système d'IA (art. 13, 3., (b) iv). Ceci permet de prendre en

compte certains type de biais, donc de discriminations spécifiques même en l'absence de définitions normatives. Des indicateurs statistiques de biais devenus relativement consensuels dans la communauté académique sont proposés dans la section suivante.

En revanche, il est regrettable qu'aucune indication, recommandation, contrainte, ne vienne ensuite préciser ce qui pourrait ou devrait être fait pour atténuer ou éliminer un biais discriminatoire. Ceci est laissé au libre arbitre du concepteur d'un système d'IA en espérant que les choix opérés soient explicitement détaillés en toute transparence pour le fournisseur qui en assume la responsabilité et pour l'utilisateur en relation avec les usagers. L'exemple numérique illustre une telle démarche.

Utilisateur & Usager Le règlement traite en priorité les considérations commerciales, donc des risques de défaillance inhérents de l'acquisition des données à la mise en exploitation d'un système d'IA. Tout système doit satisfaire aux exigences de performance annoncées selon un principe de sécurité des produits ou responsabilité du fait des produits défectueux. En revanche, l'usager final, les dommages auxquels il peut être confronté, ne sont pas du tout pris en compte. L'obligation d'information (art. 13) est ainsi au profit de l'utilisateur et pas à celui de l'usager, personne physique impactée, qui ne semble donc protégé à ce jour que par les seules obligations de l'article 22 du RGPD. Il est informé de l'usage d'un système d'IA le concernant, il peut en contester la décision auprès de l'utilisateur humain mais l'explication de la décision, des risques encourus, sont soumises aux compétences et à la déontologie professionnelle de cet utilisateur: conseiller financier pour un client, magistrat pour un justiciable, responsable des ressources humaines pour un candidat, à moins d'un cadre juridique spécifique (*e.g.* code de santé public).

Données le règlement reconnaît le rôle prépondérant des algorithmes d'apprentissage automatique et donc de la nécessité absolue (considérant 44) de qualité et pertinence des données conduisant à leur entraînement. L'article 10 impose en conséquence des compétences en Statistique pour conduire les études préalables à l'entraînement d'un algorithme. Nous assistons à un renversement de tendance, un retour de balancier, du tout automatique à une approche raisonnée sous responsabilité humaine de cette phase d'analyse des données longue et coûteuse mais classique du métier de statisticien.

Responsabilité De façon générale, l'objectif essentiel n'est plus la performance absolue comme dans les concours de type *Kaggle* et conduisant à des empilements inextricables d'algorithmes opaques mais de satisfaire à un ensemble de contraintes pour la mise en conformité, dont celle de transparence, sous la responsabilité du fournisseur du système d'IA. L'analyse des responsabilités en cas de défaillance ou de produit défectueux sera l'objet d'un autre texte.

Documentation Tous les choix opérés lors de la conception d'un système d'IA: ensembles de données, algorithmes, procédures d'apprentissage et de tests, optimisations des paramètres, compromis entre confidentialité, performances, interprétabilité, biais... doivent (art. 11 et annexe IV) être explicitement documentés en vue d'un audit *ex-ante* des systèmes de l'annexe II ou d'un contrôle *ex-post* d'un système de l'annexe III. C'est un renversement de la charge de preuve sous la responsabilité du fournisseur qui doit pouvoir montrer que le concepteur a mis en œuvre ce qui était techniquement possible pour satisfaire aux obligations (conformité) légales de sécurité, transparence, performances et non discrimination.

Autorité notifiante (Chapitre 4 Titre III) Chaque pays va se doter ou désigner (art. 30) un service chargé entre autres de superviser l'audit *ex-ante* d'un système d'IA à haut risque de l'annexe II avant son déploiement qu'il soit commercialisé ou non. L'autorité notifiante désigne l'*organisme de notification* qui exécutera l'audit. De façon assez étonnante, un système

d'ascenseur élémentaire, n'embarquant qu'une "IA" logique rudimentaire mais dépendant de l'annexe II, est plus contraint par l'obligation de certification par un organisme tiers, au contraire d'applications des systèmes d'IA de l'annexe III (justice, emploi, crédit...) impactant directement des personnes physiques avec des risques réels envers les droits fondamentaux. Il faudra donc être attentif à l'interprétation que fera un État membre de cette situation afin d'évaluer les possibilités de saisine et compétences de contrôle d'un système à haut risque de l'annexe III.

Archivage & confidentialité Le règlement cible donc, en première lecture, les obligations commerciales du fournisseur plutôt que celles étiques ou déontologiques envers l'usager. Néanmoins le règlement apporte la possibilité de prendre en compte des données sensibles (art. 10, 5.), les obligations d'archivage des décisions (art. 12), de suivi des performances selon les groupes (art. 13), une surveillance humaine (art. 14) pendant toute la période d'utilisation et de correction rétro-active des biais (art. 15). Cette obligation d'archivage et surveillance du fonctionnement notamment à destination des groupes sensibles oblige implicitement à l'acquisition, en toute sécurité (cryptage, anonymisation, pseudonymisation...), de données confidentielles (*e.g.* origine ethnique). Cela ne rend-il pas indispensable, selon le domaine d'application, la mise en place d'un protocole explicite de consentement libre et éclairé, d'un engagement éthique, entre l'utilisateur et l'usager, protégé par le RGPD? Comment sont évalués les risques encourus d'un usager ou groupe d'usager par le recueil et l'exploitation de leurs données sensibles lors de l'exploitation d'un système d'IA face aux bénéfices attendus pour eux mêmes ou l'intérêt public?

3 Prise en compte méthodologique de l'*AI Act*

3.1 Quels algorithmes

Dans l'attente d'une adoption effective du texte final qui risque d'être amendé, il est néanmoins prudent, compte tenu des investissements en jeu, d'anticiper des réponses techniques à certaines contraintes ou obligations faites aux système d'IA désignés à haut risque. Cet article laisse volontairement de côté certaines classes d'algorithmes mentionnées ou non dans l'annexe I dont la liste finale reste l'objet de débats entre les instances européennes.

Un système expert est l'association d'une base de règles logiques ou base de connaissances construites par des experts du domaine concerné, d'un moteur d'inférence et d'une base de faits observés pour une exécution en cours. Le moteur d'inférence recherche la séquence de règles logiquement applicables à partir des faits observés de la base qui s'incrémente comme conséquences du déclenchements des règles. Le processus itère jusqu'à l'obtention ou non d'une décision recherchée et expliquée par la séquence de règles y conduisant. Très développée dans les années 70, la recherche a marqué le pas face à un problème dit *NP-complet* c'est-à-dire de complexité algorithmique exponentielle en la taille de la base de connaissance (nombre de règles). Supplantée par la ré-émergence des réseaux de neurones (années 80) puis plus largement par l'apprentissage automatique, la recherche dans ce domaine dit d'IA symbolique est restée active. Elle connaît un renouveau motivé par les capacités d'explicabilité des systèmes experts.

Les approches statistiques bayésiennes ou non basées sur des données sont associées implicitement aux méthodes par apprentissage. En revanche, les algorithmes d'allocation optimale de ressources prennent une place à part. Si les principes d'allocation en tant que tels ne soulèvent pas de problème, ceux d'ordonnancement ou de tri des ressources peuvent amener des risques réels de discrimination indirecte. C'est notamment le cas de l'algorithme ParcourSup lorsque les établissements d'enseignement supérieur introduisent des pondérations selon le lycée d'origine des candidats: lycée de centre ville *vs.* lycée de

banlieue. Cette situation rejoint alors le cas des algorithmes déterministes ou procéduraux. Il s'agit d'algorithmes décisionnels (*e.g.* calcul de taxes, impôts, allocations ou prestations sociales,...) basés sur un ensemble de règles de décision déterministes qui peuvent tout autant présenter des impacts, désavantages ou risques de discrimination indirecte, malgré une apparente neutralité. La Défenseure des Droits (2020) est très attentive en France à l'analyse et détection de ces risques. Celle-ci relèvent de l'analyse experte des règles de décisions codées dans l'algorithme qui en l'état ne sont pas concernés par le projet de règlement. Néanmoins, la complexité de l'algorithme peut être telle qu'une analyse experte *ex-post* ne sera pas en mesure d'évaluer l'étendue des risques indirects. Aussi, un algorithme déterministe complexe peut être analysé avec les mêmes outils statistiques que ceux adaptés à un algorithme d'apprentissage automatique.

Nous insistons donc tout particulièrement sur les systèmes d'IA basés sur des algorithmes d'apprentissage supervisé ou statistique ou IA empirique par opposition à l'IA dite symbolique des systèmes experts. Ce sont très majoritairement les plus répandus au sein de ceux désignés à haut risque (art. 6) car susceptibles d'impacter directement des personnes physiques.

Même sans obligation de certification *ex-ante* par un organisme notifié, une documentation exhaustive (art. 11) d'un système d'IA à haut risque doit être produite et fournie à l'utilisateur. Cette section propose quelques indications méthodologiques pour répondre à cette attente.

3.2 Les données

Tout système d'IA basé sur un algorithme d'apprentissage statistique nécessite la mise en place d'une base de données d'entraînement fiable et représentative du domaine d'application visé qui doit en tout premier lieu satisfaire aux exigences de confidentialité du RGPD. Puis, le travail d'exploration statistique, généralement long et fastidieux d'acquisition, vérification, analyse, préparation, nettoyage, enrichissement, archivage sécurisé des données, est essentiel à l'élaboration d'un système d'IA performant, robuste, résilient et dont les biais potentiels sont sous contrôle. Construire de nouvelles caractéristiques (*features*) adaptées à l'objectif, traquer et gérer éventuellement par imputation des données manquantes, identifier les anomalies ou valeurs atypiques (*outliers*) sources de défaillances, les sources de biais: classes ou groupes sous représentés, biais systémiques, nécessitent compétences et expériences avancées en Statistique.

Ces compétences sont indispensables pour répondre aux attentes de l'article 10 ainsi qu'aux besoins de la documentation (annexe IV) imposée par l'article 11.

3.3 Qualité, précision et robustesse

Les articles 13 et 15 imposent clairement de devoir documenter les performances et risques d'erreur, éventuellement en fonction de groupes sensibles et protégés, ou de défaillance d'un système d'IA. Cela rend indispensable l'explicitation de choix notamment des métriques utilisées.

Choix de métrique

L'évaluation de la qualité d'une aide algorithmique à la décision est essentielle à la justification du déploiement d'un système d'IA au regard de sa balance bénéfice / risques. Dans le cas d'un système IA empirique ou par apprentissage automatique, il s'agit d'estimer la *précision* des prévisions dont les mesures sont bien connues et maîtrisées, parties intégrante du processus d'apprentissage. Néanmoins parmi un large éventail des possibles, le choix, précisément justifié, doit être adapté au domaine, au type de problème traité, aux risques spécifiques encourus quelque soit le modèle ou le type d'algorithme d'apprentissage utilisé. Citons par exemple les situations de:

Régression ou modélisation et prévision d'une variable cible Y quantitative. Elle est généralement basée sur l'optimisation d'une mesure quadratique

(norme L_2) pouvant intégrer, à l'étape d'entraînement, différents types de pénalisation dont celles de parcimonie (*ridge, Lasso*) afin de contrôler la complexité de l'algorithme et éviter les phénomènes de sur-apprentissage. D'autres types de fonction objectif basée sur une perte en norme L_1 ou valeur absolue, moins sensible à la présence de valeurs atypiques (*outliers*) que la norme quadratique, permet des solutions plus robustes car tolérantes à des observations atypiques.

Classification ou modélisation, prévision d'une variable Y qualitative. Le choix d'une mesure d'erreur doit être opéré parmi de très nombreuses possibilités: taux d'erreur, AUC (*area under the ROC Curve* pour une variable Y binaire), score F_β, risque bayésien, entropie... avec la difficile prise en compte des situations de classes déséquilibrées qui oriente le choix du type de mesure et nécessite des précautions spécifiques dans l'équilibrage de la base d'apprentissage ou les pondérations de la fonction objectif en prenant en compte une matrice de coûts de mauvais classement éventuellement asymétrique.

Limites de la précision

Besse (2021) rappelle que les performances de l'IA sont largement surévaluées par le battage médiatique dont bénéficient ces technologies. Ces performances sont d'autant plus dégradées lorsque la décision concerne la prévision d'un comportement (achat, départ, embauche, acte violent, pathologie...) individuel humain dépendant potentiellement d'un très grand nombre de variables explicatives ou facteurs dont certains peuvent ne pas être observables. Il importe de distinguer les systèmes d'IA développés dans un domaine bien déterminé (*e.g.* process industriel sous-contrôle), où le nombre de facteurs ou dimensions est raisonnable et identifié, des systèmes d'IA où opère le fléau ou malédiction de la dimension (*curse of dimensionality*), lorsque celle-ci est très grande, voire indéterminée.

L'histoire de la littérature statistique puis d'apprentissage automatique peut être lue comme une succession de stratégies pour le contrôle du nombre de variables et ainsi de paramètres estimés dans un modèle statistique ou entraînés dans un algorithme. Il s'agit par exemple de contrôler le conditionnement d'une matrice en régression: PLS (*partial least square*), sélection de variables (critères AIC, BIC), pénalisations *ridge* ou *Lasso*, et ainsi l'explosion de la variance des prévisions. Plus généralement c'est aussi le contrôle du risque de sur-ajustement qui doit être documenté comme résultat de l'optimisation des hyperparamètres: nombre de plus proches voisins, pénalité en machines à vecteurs supports, nombre de feuilles d'un arbre, de variables tirées aléatoirement dans une forêt d'arbres, profondeur des arbres et nombre d'itérations en *boosting* ... structures des couches convolutionnelles et *drop out* des réseaux de neurones en reconnaissance d'images. Même si les stratégies d'optimisation de ces hyperparamètres par validation croisée ou échantillon de validation sont bien rodées, le fléau de la dimension peut s'avérer rédhibitoire (*e.g.* Verzelen 2012).

Échantillon test

En tout état de cause, il est indispensable de mettre en place une démarche très rigoureuse pour conduire à l'évaluation de la précision et donc des performances d'un système d'IA basé sur un algorithme d'apprentissage. Comme énoncé dans l'article 3, 31. ce sont des *données de test indépendantes* de celles d'apprentissage qui sont utilisées à cet effet. *Attention* néanmoins d'évaluer les performances sur des données telles qu'elles se présenteront *réellement* en exploitation, avec leurs défauts, et pas un simple sous-ensemble aléatoire de la base d'apprentissage comme c'est trop souvent le cas en recherche académique. En effet cet ensemble de données peut bénéficier d'une homogénéité d'acquisition (*e.g.* même technologie, même opérateur) et de prétraitements qui peuvent faire défaut à de réelles données d'entrée à venir en exploitation. Cela demande donc une extrême rigueur dans la constitution

d'un échantillon test pour éviter ces pièges bien trop présents en recherche académique (*e.g.* Liu et al. 2019, Roberts et al. 2021) et conduisant, sous la pression de publication, à beaucoup trop de résultats non reproductibles et des algorithmes non certifiables. Enfin, une surveillance (art. 14) toute la durée de vie du système d'IA est indispensable afin d'en détecter de possibles dérives ou dysfonctionnements (art. 12 et 15) affectant la robustesse ou la résilience des décisions.

Robustesse

L'évaluation de la *robustesse* est liée aux procédures de contrôle mises en place pour détecter des valeurs atypiques (*outliers*) ou anomalies dans la base d'apprentissage et au choix de la fonction perte de la procédure d'entraînement de l'algorithme. Impérativement, surtout dans les d'applications sensibles pouvant entraîner des risques élevés en cas d'erreur, la détection d'anomalie doit également être intégrée en exploitation afin de ne pas chercher à proposer des décisions correspondant à des situations atypiques, étrangères à la base d'apprentissage.

Résilience

La *résilience* d'un système d'IA est essentielle pour les dispositifs critiques (dispositifs de santé connecté, aide au pilotage). Cela concerne par exemple la prise en compte de données manquantes lors de l'apprentissage comme en exploitation. Il s'agit d'évaluer la capacité d'un système d'IA à assurer des fonctions pouvant s'avérer vitales en cas, par exemple, de panne ou de fonctionnement erratique d'un capteur: choix d'un algorithme tolérant aux données manquantes, imputation de celles-ci, fonctionnement en mode dégradé, alerte et arrêt du système.

3.4 Explicabilité

Une recherche active

Il est bien trop tôt pour tenter un résumé opérationnel de ce thème et fournir des indications claires sur la démarche à adopter pour satisfaire aux exigences réglementaires (art. 13, 15). Il faut pour cela attendre que la recherche ait progressé et qu'une sélection "naturelle" en extrait les procédures les plus pertinentes parmi une grande quantité de solutions proposées; un article de revue sur ce sujet (Barredo Arrieta et al. 2020) listait plus de 400 références.

Arbre de choix

Tentons de décrire les premiers embranchements d'un arbre de décision en répondant à quelques questions rudimentaires qu'il faudrait en plus adapter au domaine d'application car le type de réponse à apporter n'est évidemment pas le même s'il s'agit d'expliquer le refus d'un prêt ou les conséquences d'une aide automatisée au diagnostic d'un cancer.

Il importe de bien distinguer les niveaux d'explication: concepteur, utilisateur ou usager, même si ce dernier n'est pas directement concerné par le projet de règlement. De plus, l'explication peut s'appliquer soit au fonctionnement général de l'algorithme soit à une décision spécifique.

Il y a schématiquement deux types d'algorithmes dontceux relativement transparents: modèles linéaires et arbres de décision. L'explication est dans ce cas possible à condition que le nombre de variables et d'interactions prises en compte ou le nombre de feuilles d'un arbre reste raisonnable. Toutes les autres classes d'algorithme d'apprentissage, systématiquement non linéaires et complexes, sont par construction opaques. Il s'agit alors de construire une explication par différentes stratégies comme une approximation explicable par un modèle linéaire, un arbre ou un ensemble de règles de décision déterministes. Une autre stratégie consiste à fournir des indications sur l' *importances des variables* en mesurant l'effet d'une permutation aléatoire de leurs valeurs (*mean decrease accuracy* Breiman, 2001), en stressant l'algorithme (Bachoc

et al. 2020) ou en réalisant une analyse de sensibilité par indices de Sobol (Bénesse et al. 2021).

Le concepteur d'un algorithme s'intéresse également à l'explication d'une décision spécifique afin d'identifier la cause d'une erreur, y remédier par exemple en complétant la base d'apprentissage d'un groupe sous-représenté avant de ré-entraîner l'algorithme. L'utilisateur d'un système d'IA doit être au mieux informé (art. 13, 15) des possibilités d'expliquer une décision qu'il pourra retranscrire à l'usager (client, patient, justiciable, citoyen...) selon sa propre déontologie, son intérêt commercial ou une contrainte légale par exemple pour des décisions administratives. Pour ce faire quelques stratégies sont proposées comme une *approximation locale* par un modèle explicable (linéaire, arbre de décision) ou par une liste d'exemples *contrefactuels* c'est-à-dire des situations les plus proches, en un certain sens, qui conduiraient à décision contraire, généralement plus favorable (attribution d'un prêt). Lorsque cela s'avère impossible, comme par exemple dans le cas d'un diagnostic médical impliquant un nombre important de facteurs opaques, il importe d'informer précisément l'utilisateur et donc le patient sur les risques d'erreur afin que consentement de ce dernier soit effectivement libre et éclairé.

Quelques démonstrations de procédures explicatives sont proposées sur des sites en accès libre. Citons: `gems-ai.com`, `aix360.mybluemix.net`, `github.com/MAIF/shapash`

Réalité complexe

Ne pas perdre de vue que l'impossibilité ou simplement la difficulté à formuler une explication provient certes de l'utilisation d'algorithmes opaques mais dont la nécessité est inhérente à la complexité même du réel. Un réel complexe (*e.g.* les fonctions du vivant) impliquant de nombreuses variables, leurs interactions, des effets non linéaires voire des boucles de contre-réaction, est nécessairement modélisé par un algorithme complexe afin d'éviter des simplifications abusives pouvant gravement nuire aux performances. C'est tout d'abord le réel qui s'avère complexe à expliquer.

3.5 Biais & discrimination

Bien que très présente dans les textes préliminaires (livre blanc (CE 2021, considérants de l'AI Act) la référence au risque de discrimination ne l'est pas de façon explicite dans les projets d'articles. Apparaissent néanmoins l'obligation de détecter des biais dans les données (art. 10) ainsi que celle d'afficher des performances ou risques d'erreur par groupe (art. 13). Quelles en sont les conséquences au regard des difficultés de définir, détecter une discrimination qu'elle soit humaine ou algorithmique?

Détecter une discrimination

Formellement, la stricte équité peut s'exprimer par des propriétés d'indépendance en probabilité entre la variable cible Y qui exprime une décision et la variable dite sensible S par rapport à laquelle une discrimination est en principe interdite. Cette variable peut être quantitative (*e.g.* âge) ou qualitative à deux ou plusieurs classes (*e.g.* genre ou origine ethnique) ou, de façon plus complexe, la prise en compte d'interactions entre plusieurs variables sensibles. Néanmoins cette définition théorique de l'équité n'est pas concrètement praticable pour détecter, mesurer, atténuer des risques de biais. De plus, les textes juridiques font essentiellement référence à un groupe de personnes sensibles par rapport aux autres. En conséquences et pour simplifier cette première lecture pédagogique de la détection des risques de discrimination, nous ne considérons qu'une variable sensible à 2 modalités: jeune *vs.* vieux, femme *vs.* homme...

Une façon bien établie de détecter une décision humaine discriminatoire consiste à opérer par *testing*. Dans le cas d'une présomption de discrimination à l'embauche, la procédure consiste à adresser deux CV comparables, à l'exception (*counterfactual example*) de la modalité de la variable sensible (*e.g.* genre, origine ethnique associée au nom) afin de comparer les réponses:

proposition ou non d'entretien. Cette démarche individuelle est rendue systématique (Rich, 2014) dans une enquête par l'envoi de milliers de paires de CV. C'est en France la doctrine officielle promue par le Comité National de l'Information Statistique et commanditée périodiquement par la DARES (Direction de l'Animation, des Études, de la Recherche et des Statistiques) du Ministère du travail.

Des indicateurs statistiques peuvent être estimés à l'issue de cette enquête mais, comme il n'existe pas de définition juridique de l'équité qui devient par défaut l'absence de discrimination, le monde académique a proposé quelques dizaines d'indicateurs (*e.g.* Zliobaitė 2017) afin d'évaluer des biais potentiels sources de discrimination. Il est nécessaire d'opérer des choix parmi tous les critères de biais en remarquant que beaucoup de ces indicateurs s'avèrent être très corrélés ou redondants (Friedler et al. 2019). Empiriquement et après avoir consulter une vaste littérature sur l'IA éthique ou plutôt sur les risques identifiés de discrimination algorithmique, un consensus émerge sur le choix en priorité de trois niveaux de biais statistique. Sont finalement considérés dans cet article élémentaire trois types de rapports de probabilités (égaux à 1 en cas d'indépendance stricte) dont Besse et al. (2021) proposent des estimations par intervalle de confiance afin d'en contrôler la précision.

Parité statistique et effet disproportionné

Le premier niveau de risque de discrimination algorithmique s'illustre simplement: si un algorithme est entraîné sur des données biaisées, il apprend et reproduit très fidèlement ces biais systémiques, de société ou de population, par lesquels un groupe est historiquement (*e.g.* revenu des femmes) désavantagé; plus grave, l'algorithme risque même de renforcer le biais en conduisant à des décisions explicitement discriminatoires. Il importe donc de pouvoir détecter, mesurer, atténuer voire éliminer ce type de biais. L'équité ou parité statistique (ou *demographic equality*) serait l'indépendance entre la ou les variables sensibles S (*e.g.* genre, origine ethnique) et la variable de prévision $\widehat{Y}$ de la décision. Historiquement, l'écart à l'indépendance pour mesurer ce type de biais est évalué aux USA dans les procédures d'embauche depuis 1971 par la notion d'effet disproportionné ou *disparate impact* et maintenant reprises systématiquement (Barocas et Selbst, 2016) pour l'évaluation de ce type de discrimination dans un algorithme. L'effet disproportionné consiste à estimer le rapport de deux probabilités: probabilité d'une décision favorable ($\widehat{Y}=1$) pour une personne du groupe sensible ($S=0$) au sens de la loi sur la même probabilité pour une personne de l'autre groupe ($S=1$):

$$DI = \frac{\mathbb{P}(\widehat{Y}=1|S=0)}{\mathbb{P}(\widehat{Y}=1|S=1)}.$$

Cet indicateur est intégré au *Civil Rights act & Code of Federal Regulations (Title 29, Labor: Part 1607 Uniform guidelines on employee selection procedures)* depuis 1978 avec la règle dite des 4/5 ème; si DI est inférieur à $0,8$, l'entreprise doit en apporter les justifications économiques. Les logiciels commercialisés aux USA et proposant des algorithmes de pré-recrutement automatique anticipent ce risque juridique (Raghavan et al. 2019) en intégrant une procédure automatique d'atténuation du biais (*fair learning*). Il n'y a aucune obligation ni mention en France de cet indicateur statistique, seulement une incitation de la part de la Défenseure des Droits et de la CNIL (2012) envers les services de ressources humaines des entreprises. Il leur est suggéré de tenir des statistiques ethniques, autorisées dans ce cas sous réserve de confidentialité, sous la forme de tables de contingence dont il serait facile d'en déduire des estimations d'effet disproportionné.

La mise en évidence d'un biais systémique est implicitement citée lors de l'étape d'analyse préliminaire des données (art. 10, 2., (f)) mais sans plus de précision sur la façon dont il doit être pris en compte alors que renforcer algorithmiquement ce biais serait ouvertement discriminatoire. De plus serait-il politiquement opportun d'introduire une part de discrimination positive afin d'atténuer la discrimination sociale? C'est évoqué dans le guide des experts

(CE, 2019, ligne directrice 52) pour *améliorer le caractère équitable de la société* et techniquement l'objet d'une vaste littérature académique nommée apprentissage équitable (*fair learning*). Cette opportunité n'est pas reprise explicitement dans l'*AI Act* mais nous verrons dans l'exemple numérique ci-dessous qu'elle ne peut être exclue et peut même être pleinement justifiée en prenant en considération les autres types de biais ci-après.

Erreurs conditionnelles

Les taux d'erreur de prévision et donc les risques d'erreur de décisions sont-ils les mêmes pour chaque groupe (*overall error equality*)? Autrement dit, l'erreur est-elle indépendante de la variable sensible? Ceci peut se mesurer par l'estimation (intervalle de confiance) du rapport de probabilités (probabilité de se tromper pour le groupe sensible sur la probabilité de se tromper pour l'autre groupe):

$$REC = \frac{\mathbb{P}(\widehat{Y} \neq Y|S=0)}{\mathbb{P}(\widehat{Y} \neq Y|S=1)}.$$

Ainsi, si un groupe est sous-représenté dans la base d'apprentissage, il est très probable que les décisions le concernant soient moins fiables. C'est une des première critiques formulées à l'encontre des algorithmes de reconnaissance faciale et ce risque est également présent dans les applications en santé (Besse et al. 2020) ou en ressources humaines (De-Arteaga et al. 2019). L'identification, la prise en compte et la surveillance de ce risque sont présents (art. 13, 3., (b), ii et art. 15, 1. & 2.) dans le projet de règlement et doivent donc être explicitement détaillés dans la documentation (art. 11).

Rapports de cote conditionnels

Même si les deux critères précédents sont trouvés équitables, les erreurs peuvent être dissymétriques (plus de faux positifs, moins de faux négatifs) au détriment d'un groupe avec un impact d'autant plus discriminatoire que le taux d'erreur est important. Cet indicateur (comparaison des rapports de cote ou *odds ratio* d'indépendance conditionnelle nommé aussi *equalli odds*) est au cœur de la controverse concernant l'évaluation COMPAS du risque de récidive aux USA (Larson et al. 2016). Il est également présent dans l'exemple numérique ci-après. Cet indicateur double est mesuré par l'estimation de deux rapports de probabilités: rapports des taux de faux positifs du groupe sensible sur le taux de faux positifs de l'autre groupe et rapport des taux de faux négatifs pour ces mêmes groupes.

$$RFP = \frac{\mathbb{P}(\hat{Y}=1|Y=0,S=0)}{\mathbb{P}(\hat{Y}=1|Y=0,S=1)} \quad \text{et} \quad RFN = \frac{\mathbb{P}(\widehat{Y}=0|Y=1,S=0)}{\mathbb{P}(\widehat{Y}=0|Y=1,S=1)}.$$

L'évaluation de ce type de biais n'est pas explicitement mentionné dans le projet de règlement. Néanmoins il fait partie de la procédure classique d'évaluation des erreurs en classification à l'aide d'une matrice de confusion ou de courbes ROC par groupes et ne peut être négligé.

Notons qu'il est d'autant plus difficile de faire abstraction du dernier type de biais que les trois sont interdépendants et même en interaction avec les autres risques: précision et explicabilité. Ceci est clairement mis en évidence dans l'exemple numérique suivant. Il y a donc une forme d'obligation déontologique ou de cohérence statistique à devoir appréhender ces différents niveaux d'analyse.

4 Exemple numérique

L'exemple jouet ou bac à sable de cette section permet d'illustrer concrètement toute la complexité des principes précédemment évoqués en soulignant leur interdépendance. Ce jeu de données est ancien, largement utilisé pour illustrer tous les travaux visant une atténuation optimale du biais. Le monde académique espère avoir rapidement accès à bien d'autres "bac à sable" représentatifs dont la construction est l'objet de l'article 53 de l'*AI Act*.

4.1 Données

Les données publiques utilisées imitent le contexte du calcul d'un score de crédit. Elles sont extraites (échantillon de 45 000 personnes) d'un recensement de 1994 aux USA et décrivent l'âge, le type d'emploi, le niveau d'éducation, le statut marital, l'origine ethnique, le nombre d'heures travaillées par semaine, la présence ou non d'un enfant, les revenus ou pertes financières, le genre et le niveau de revenu bas ou élevé. Elles servent de référence ou *bac à sable* pour tous les développements d'algorithmes d'apprentissage automatique équitable. Il s'agit de prévoir si le revenu annuel d'une personne est supérieur ou inférieur à 50k$ et donc de prévoir, d'une certaine façon, sa solvabilité connaissant ses autres caractéristiques socio-économiques. Ces questions de discrimination dans l'accès au crédit sont toujours d'actualité (Campisi 2021, Hurlin et al. 2021, Kozodoi et al. 2021) même si le principe du *score de crédit* s'est généralisé dès les années 90 avec l'envol du *data mining* devenu depuis de l'IA.

L'étude complète et les codes de calcul sont disponibles dans un tutoriel (calepin *Jupyter*) mais l'illustration est limitée à un résumé succinct de l'analyse de la discrimination selon le genre.

4.2 Résultats

Une analyse exploratoire: nettoyage des données, description statistique, préalable doit être incluse dans la documentation. Elle est l'objet d'un autre tutoriel dont seuls quelques résultats sont retenus par souci de concision. Ils mettent en évidence un biais systémique ou de société important: seulement $11,6\%$ des femmes ont un revenu élevé contre $31,5\%$ des hommes. Le rapport $DI = 0,38$ est donc très disproportionné et peut s'expliquer par quelques considérations sociologiques bien identifiées sur le premier plan factoriel (fig. 12.1) d'une analyse factorielle multiple des correspondances calculée après avoir recodé qualitatives toutes les variables. Les femmes travaillent en moyenne moins d'heures (`HW1`) par semaine (occupations ménagères et enfants?); même si le niveau de diplôme ne semble pas lié au genre, elles occupent un poste avec moins de responsabilité (`Admin`) (effet plafond de verre?). Un autre type de biais semble présent dans ces données, les femmes sont associées (co-occurrences plus fréquentes que l'indépendance) à la présence d'enfants sans pour autant être en situation de couple contrairement aux hommes. Cette enquête s'adresse-t-elle de façon privilégiée au chef ou à la cheffe de famille éventuellement monoparentale?

Les données ont été aléatoirement réparties en trois échantillons d'apprentissage (29 000), destinés à l'estimation des modèles ou entraînement des algorithmes, de validation (8000) afin d'optimiser certains hyper paramètres et de test (8000) pour évaluer les différents indicateurs de performance et biais. La taille relativement importante de l'échantillon initial permet de considérer un échantillon de validation représentatif, comme demandé dans le règlement, afin d'éviter des procédures plus lourdes de validation croisée. Les résultats de prévision sont regroupés dans la figure 12.2.

Le biais systémique (`dataBaseBias`) des données est comparé avec celui de la prévision de niveau de revenu par un modèle classique linéaire de régression logistique `linLogit`: $DI = 0,25$. Significativement moins élevé (intervalles de confiance disjoints), il montre que ce modèle renforce le biais et donc discrimine nettement les femmes dans sa prévision. La procédure naïve (`linLogit-w-s`) qui consiste à éliminer la variable dite sensible (genre) du modèle ne supprime en rien $(DI = 0,27)$ le biais discriminatoire car le genre est de toute façon présent à travers les valeurs prises par les autres variables (effet *proxy*). Une autre conséquence de cette dépendance aux proxys est que le *testing* ou *counterfactual test* (changement de genre toutes choses égales par ailleurs) ne détecte plus $(DI = 0,90)$ aucune discrimination!

Un algorithme non-linéaire élémentaire (`tree`, arbre binaire de décision) augmente le biais mais pas de façon statistiquement significative car les intervalles de confiance ne sont pas disjoints. Sa précision est meilleure que celle du modèle de régression logistique mais, si l'objectif est une interprétation utile,

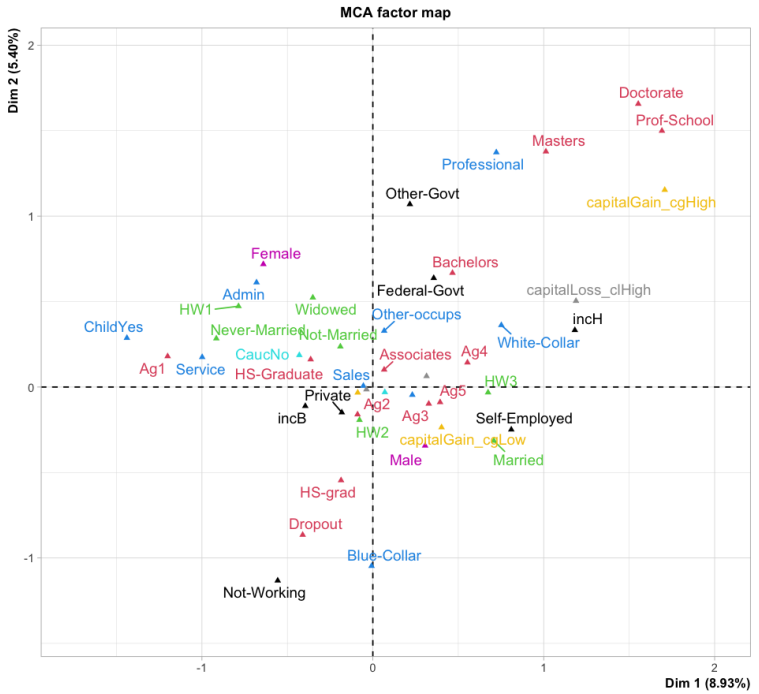


Figure 12.1: *Premier plan factoriel d'une analyse factorielle multiple des correspondances (librairie* `FactoMineR`*, Lê et al. 2008)*

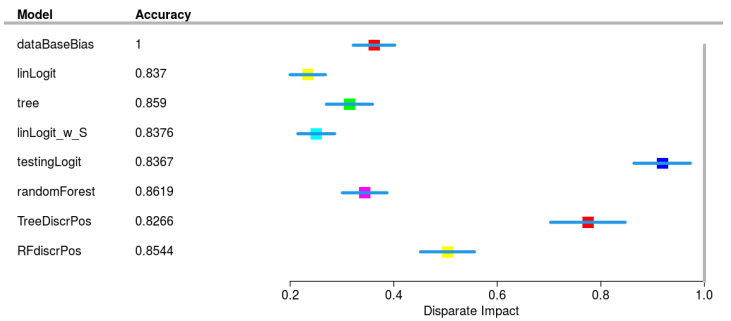


Figure 12.2: *Précision de la prévision (accuracy) et effet disproportionné (discrimination en fonction du genre) estimé par un intervalle de confiance sur un échantillon test (taille 9000) pour différents modèles ou algorithmes d'apprentissage.*

il est nécessaire de réduire la complexité de l'arbre en pénalisant le nombre de feuilles, d'une centaine à une dizaine. Dans ce cas la précision se dégrade pour rejoindre celle de la régression logistique; l'explicabilité a un coût.

Un algorithme non linéaire plus sophistiqué (`random forest`) est très fidèle au biais des données avec un indicateur ($DI = 0,36$) proche de celui du biais de société et fournit une meilleure précision: $0,86$ au lieu de $0,84$ pour la régression logistique. Cet algorithme ne discrimine pas plus, apporte une meilleure précision, mais c'est au prix de l'explicabilité du modèle. Opaque comme un réseau de neurones, il ne permet pas d'expliquer une décision à partir de ses paramètres comme cela est facile avec le modèle de régression ou un arbre binaire de décision de taille raisonnable.

Une question délicate concerne le choix politique de procéder ou non à une atténuation du biais systémique dans le cas d'un score de crédit. Contrairement à Hurlin et al. (2021), Goglin (2021) l'aborde de façon très incomplète en ne considérant, de manière exclusive, que le biais des erreurs selon le genre. Cet auteur "justifie" de ne pas considérer le biais systémique car le corriger conduirait des femmes à des situations de surendettement tandis que le 3ème type de biais est purement oublié. Une analyse plus fine montre, à travers cet exemple, toute l'importance de prendre en compte simultanément les trois types de biais afin d'éviter un positionnement quelque peu "paternaliste".

En principe, la précision de la prévision pour un groupe dépend de sa représentativité. Si ce dernier est sous-représenté, l'erreur est plus importante; c'est typiquement le cas en reconnaissance faciale mais pas dans l'exemple traité. Alors qu'elles sont deux fois moins nombreuses dans l'échantillon, le taux d'erreur de prévision est de l'ordre de $7,9\%$ pour les femmes et de 17% ($REC = 0,36$) pour les hommes (algorithme d'arbre binaire simplifié). Il est alors indispensable de considérer le troisième type de biais pour se rendre compte que c'est finalement au désavantage des femmes. Le taux de faux positifs est plus important pour les hommes $(0,081)$ que pour les femmes $(0,016)$ ($RFP = 0,20$). Ceci avantage les hommes qui bénéficient plus largement d'une décision favorable même à tort. En revanche, le taux de faux négatifs est plus important pour les femmes $(0,41)$, à leur désavantage, que pour les hommes $(0,38)$ ($Rfn = 1,08$) mais ces dernières différences ne sont pas significatives.

Dans une telle situation en choisissant le seuil de décision par défaut à $0,5$, une banque prendrait peu de risque: faible taux de faux positifs et taux élevés de faux négatifs mais, conclusion importante, il apparaît une *rupture d'équité* au sens où la banque prend *plus de risques au bénéfice des hommes* alors que les taux d'erreur les concernant sont plus élevés.

Une atténuation du biais des rapports de cotes se justifie donc afin de rendre comparables les chances d'obtention d'un crédit selon le genre et ce même à tort. Plutôt que d'équilibrer ces chances en pénalisant celles des hommes, une part de discrimination positive est introduite au bénéfice des femmes pour plus d'équité en cherchant à rendre égaux les taux de faux positifs selon le genre et évalués sur l'échantillon de validation.

Les deux dernières lignes de la figure 12.2 proposent une façon simple (*post-processing*), parmi une littérature très volumineuse, de corriger le biais pour plus de *justice sociale*. Deux algorithmes sont entraînés, un par genre et le seuil de décision (revenu élevé ou pas, accord ou non de crédit...) est abaissé pour les femmes : $0,3$ pour les forêts aléatoires, $0,2$ pour un arbre binaire, au lieu de celui par défaut de $0,5$ pour les hommes. Cette correction des faux positifs impacte également les taux d'erreur qui deviennent plus équilibrés selon le genre et provoque également une atténuation de l'effet disproportionné pour une *société plus équitable*. L'arbre binaire utilisé (`TreeDiscrPos`) est celui pénalisé (peu de feuilles) afin d'obtenir une interprétation facile au prix de la précision. Les seuils et le paramètre de pénalisation ont été déterminés sur l'échantillon de validation avant d'être appliquées indépendamment à l'échantillon test.

4.3 Discussion

Nous pouvons tirer quelques enseignements de cet exemple jouet imitant le calcul d'un score d'attribution de crédit bancaire.

- Sans précaution, si un biais est présent dans les données, il est appris et même renforcé par un modèle linéaire élémentaire.
- La suppression naïve de la variable sensible (genre) pour réduire le biais n'y change rien d'où l'importance (art. 10, 5.) d'autoriser la prise d'un risque contrôlé de confidentialité pour intégrer des données personnelles sensibles afin de pouvoir détecter des biais.
- Un algorithme sophistiqué, non linéaire et impliquant les interactions entre les variables, ne fait que reproduire le biais mais, opaque, ne permet plus de justification des décisions si l'effet disproportionné est juridiquement attaquable comme aux USA ($DI < 0,8$). Dans le cas présent, un simple arbre binaire pénalisé pour contrôler le nombre de feuilles permet de concilier accroissement peu important du biais et explicabilité sans trop pénaliser la précision.
- En présence de proxys du genre comme c'est le cas dans cet exemple, une procédure de *testing* (*counterfactual test*) est complètement inadaptée à la détection *ex-post* d'une discrimination algorithmique. Seule une analyse rigoureuse d'une documentation loyale (art. 11) décrivant les données, la procédure d'apprentissage, les performances, peut donc s'avérer convaincante sur les capacités non discriminatoires d'un algorithme.
- Sur cet exemple, le choix d'un *post-processing* permettant d'atténuer le biais des rapports de cotes conditionnels (taux de faux positifs similaires) selon le genre impacte les trois types de biais pour en réduire simultanément l'importance. C'est une façon de légitimer l'introduction d'une dose de discrimination positive qui réduit le désavantage fait aux femmes sans pour autant nuire aux hommes.
- Finalement dans cet exemple illustratif, un arbre pénalisé pour être suffisamment simple (nombre réduit de feuilles) et assorti d'une touche de discrimination positive fournit une aide à la décision explicable à un client et équitable en terme de risques de la banque vis-à-vis de son genre.
- Certes, dans le cas d'un score de crédit, cela aurait pour conséquence d'accroître le risque de la banque en réduisant la qualité de prévision et augmentant le taux de faux positifs pour les femmes mais lui fournirait des arguments tangibles de communication pour une image "éthique": des décisions inclusives donc plus équitables et plus explicables sans trop nuire à la précision.

5 Conclusion

Comme le rappelle Meneceur (2021-b) dans une comparaison exhaustive des démarches institutionnelles, les très nombreuses approches éthiques visant à encadrer le développement et l'application des systèmes d'IA ne sont pas des réponses suffisantes et convaincantes pour développer la confiance des usagers. Ceci motive la démarche de la CE aboutissant à la publication de ce projet de règlement alors que le *Conseil de l'Europe envisage également un mélange d'instruments juridiques contraignants et non contraignants pour prévenir les violations des droits de l'homme et des atteintes à la démocratie et à l'État de droit*; la nécessité de conformité se substitue à l'éthique.

L'analyse du projet de règlement européen montre des avancées significatives pour plus de transparence des systèmes d'IA:

- importance fondamentale des données et donc de leur analyse préalable fouillée et documentée,

- évaluation et documentation explicite des performances et donc des risques d'erreur ou de manquement: robustesse, résilience,
- documentation explicite sur les capacités d'interprétation d'un système, d'une décision, à la mesure des technologies et méthodes disponibles,
- prise en compte de certains types de biais: équité sociale dans les données, performances selon des groupes et suivi des risques possibles de discrimination associés,
- enregistrement de l'activité pour une traçabilité du fonctionnement,
- contrôle humain approprié pour réduire et anticiper les risques,
- obligation de fournir la documentation exhaustive à l'utilisateur (système d'IA de l'annexe III), qui est auditée *ex-ante* par un organisme notifié pour les systèmes d'IA de l'annexe II, pour l'obtention du marquage "CE".

Néanmoins ce projet de règlement principalement motivé par une harmonisation des relations commerciales au sein de l'Union selon le principe de sécurité des produits ou de la responsabilité du fait des produits défectueux ne prend pas en compte des dommages pouvant impacter les usagers. Les conséquences ou objectifs de la démarche adoptée par la CE rejoignent d'ailleurs les exigences de la FTC (*Federal Trade Commission*) (Jillson, 2021) de loyauté et transparence vis-à-vis des performances d'un système d'IA commercialisé. Aussi certains droits fondamentaux, bien que retenus comme *exigence essentielle* dans le livre blanc se trouvent pour le moins négligés et ce d'autant plus que les systèmes d'IA à haut risque de l'annexe III ne sont pas concernés par la certification d'un organisme notifié indépendant.

- Plus largement que les seules applications de l'IA, une prise en compte d'une forme de frugalité numérique afin de réduire les impacts environnementaux ne semblent pas, dans ce projet d'*AI Act*, une préoccupation majeure de la CE. Cela concerne la consommation énergétique pour le stockage massif et l'entraînement des algorithmes et la sur-exploitation des ressources minières nécessaires à la fabrication des équipements numériques.

- Il est certes conseillé de rechercher des biais potentiels dans les données (art. 10, 2., (f)) avec même la possibilité de prendre en compte des données personnelles sensibles (art.10, 5.) pour traquer des biais systémiques sources potentielles de discrimination. Néanmoins, l'absence de précisions sur la façon de mesurer ces biais, de les atténuer ou les supprimer dans les procédures d'entraînement laisse un vide potentiellement préjudiciable à l'usager. Alors qu'il est déjà fort complexe pour un usager d'apporter la preuve d'une présomption de discrimination, par exemple par *testing*, lors d'une décision humaine, l'exemple numérique ci-dessus montre que c'est mission impossible face à une décision algorithmique. Seule une procédure rigoureuse d'audit de la documentation décrivant les données, la procédure d'apprentissage et les dispositions mises en place pour gérer, atténuer les biais, peut garantir une protection *a minima* des usagers finaux contre ce type de discrimination. Cette mise en conformité agit comme un renversement de la charge de la preuve mais qui ne bénéficie, pour les systèmes d'IA de l'annexe III, qu'à l'information de l'utilisateur pas, dans l'état actuel, à la protection de l'usager.

- Consciente de ces problèmes la Défenseure des Droits a récemment publié un avis en collaboration avec le réseau européen EQUINET dont les principales conclusions sont résumées dans un communiqué de presse. Elle y *appelle à replacer le principe de non-discrimination (de l'usager) au cœur du projet d'AI Act*. Une des questions essentielles reste à savoir qui pourra, en dehors de l'utilisateur, avoir accès à la documentation d'un système d'IA à haut risque, et donc de pouvoir l'auditer dans de bonnes conditions. Ce sera sans doute à chaque État membre de légiférer sur ces

questions.

- Notons que le Laboratoire Nationale de Métrologie et d'Essai (LNE) a pris les devants en proposant un référentiel de certification de processus pour l'IA (LNE 2021). Ce référentiel concerne le processus de conception d'un système d'IA et non la certification du produit final requérant la connaissance de normes encore à définir. Le LNE jouera le rôle d'organisme notifié pour les systèmes de transport de l'annexe II et sa filiale GMED pour les dispositifs de santé sous la responsabilité de l'Agence Nationale de Sécurité des Médicaments comme autorité notifiante.

- Le Conseil d'État (2022) publie un rapport dont la recommandation finale vise spécifiquement à combler certaines des lacunes identifiées de l'*AI Act*.

> L'étude préconise enfin une transformation profonde de la CNIL en autorité de contrôle nationale responsable de la régulation des systèmes d'IA, notamment publics, pour incarner et internaliser le double enjeu de la protection des droits et libertés fondamentaux, d'une part, et de l'innovation et de la performance publique, d'autre part.

L'exemple numérique jouet a également pour mérite de montrer clairement l'*interdépendance* de toutes les contraintes: confidentialité, qualité, explicabilité, équité (types de biais), que devrait satisfaire un système d'IA pour gagner la confiance des usagers. Il montre aussi que le problème ne se réduit pas à un simple objectif de minimisation d'un risque quantifiable pour l'obtention d'un meilleur compromis. C'est plutôt la recherche d'une moins mauvaise solution imbriquant des choix techniques, économiques, juridiques, politiques qu'il sera nécessaire de clairement expliciter dans la documentation rendue obligatoire par l'adoption à venir d'un *AI Act* qui serait, de toute façon et malgré les limites actuelles du projet de texte, une avancée notable pour plus de transparence.

Références

- Bachoc F., Gamboa F., Halford M., Loubes J.-M., Risser L. (2020). Entropic Variable Projection for Model Explainability and Intepretability, arXiv preprint: 1810.07924.

- Barocas S., Selbst A. (2016). Big Data's Disparate Impact, *California Law Review*, 104, 671.

- Barredo Arrieta A., Díaz-Rodríguez N., Del Ser J., Bennetot A., Tabik S., Barbado A., Garcia S., Gil-Lopez S., Molina D., Benjamins R., Chatila R., Herrera F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward Responsable AI, arXiv.

- Bénesse C., Gamboa F., Loubes J.-M., Boissin T. (2021). Fairness seen as Global Sensitivity Analysis, ArXiv, à paraître. responsible AI, *Information Fusion*, Vol. 58, pp 82-115.

- Besse P. (2021). Médecine, police, justice : l'intelligence artificielle a de réelles limites, The Conversation, 01/12/2021.

- Besse P., Besse Patin A., Castets Renard C. (2020). Implications juridiques et éthiques des algorithmes d'intelligence artificielle dans le domaine de la santé, *Statistique & Société*, 3, pp 21-53.

- Besse P., Castets-Renard C., Garivier A., Loubes J.-M. (2019). L'IA du Quotidien peut elle être Éthique? Loyauté des Algorithmes d'Apprentissage Automatique, Statistique et Société, VOl6 (3), pp 9-31.

- Besse P., del Barrio E., Gordaliza P., Loubes J-M., Risser L. (2021) A survey of bias in Machine Learning through the prism of Statistical Parity for the Adult Data Set, *The American Statistician*, DOI: 10.1080/00031305.2021.1952897, version en accès libre.
- Breiman L. (2001). Random forests, *Machine Learning* 45, 5–32.
- Campisi N. (2021). From Inherent Racial Bias to Incorrect Data—The Problems With Current Credit Scoring Models, Forbes Advisor.
- Castets Renard C., Besse P. (2022). Responsabilité ex ante de l'AI Act: entre certification et normalisation, à la recherche des droits fondamentaux au pays de la conformité, dans "Un droit de l'intelligence artificielle : entre règles sectorielles et régime général. Perspectives de droit comparé", dir. C. Castets-Renard et J. Eynard, Bruylant (à paraître).
- CE (2019) Lignes Directrices pour une IA digne de Confiance, rédigé par un groupe d'experts européens.
- CE (2020) Livre blanc sur l'intelligence artificielle: une approche européenne d'excellence et de confiance.
- CE (2021). Règlement du parlement et du conseil établissant des règles harmonisées concernant l'intelligence artificielle (législation sur l'intelligence artificielle) et modifiant certains actes législatifs de l'union.
- Conseil d'État (2022). S'engager dans l'intelligence artificielle pour un meilleur service public, rapport d'étude mis en ligne le 30/08/2022.
- De-Arteaga M., Romanov A. et al. (2019). Bias in Bios: A Case Study of Semantic Representation Bias in a High-Stakes Setting, *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pp 120–128.
- Défenseure des Droits (2020). Algorithmes: prévenir l'automatisation des discriminations, rapport.
- Défenseur des Droits, CNIL (2012). Mesurer pour progresser vers l'égalité des chances. Guide méthodologique à l'usage des acteurs de l'emploi.
- Friedler S., Scheidegger C., Venkatasubramanian S., Choudhary S., Hamilton E., Roth D. (2019). Comparative study of fairness-enhancing interventions in machine learning. *Proceedings of the Conference on Fairness, Accountability, and Transparency*, p. 329–38.
- Goglin C. (2021). Discrimination et IA : comment limiter les risques en matière de crédit bancaire, The Conversation, 23/09/2021.
- Hurlin C., Pérignon C., Saurin S. (2021) The fairness of credit score models, preprint SSRN.
- Jillson E. (2021). Aiming for truth, fairness, and equity in your company's use of AI, blog, consulté le 29/05/2021.
- Kozodoi N., Jacob, J. Lessman, S. (2021). Fairness in credit scoring: assessment, implementation and profit implications, preprint arXiv.
- Larson J., Mattu S., Kirchner L., Angwin J. (2016). How we analyzed the compas recidivism algorithm. ProPublica, en ligne consulté le 28/04/2020.
- Lê, S., Josse, J., Husson, F. (2008). FactoMineR: An R Package for Multivariate Analysis. Journal of Statistical Software. 25(1). pp. 1-18.
- Liu X., L. Faes, A. U. Kale et al. (2019), A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis, *The Lancet Digital Health*, vol. 1, pp. e271–e297.

- LNE (2021). Référentiel de Certification du Processus IA, Laboratoire Nationale de Métrologie et d'Essais.
- Meneceur Y. (2021). Analyse des principaux cadres supranationaux de régulation de l'intelligence artificielle : de l'éthique à la conformité, projet d'étude, Institut des Hautes Études sur la Justice (IHEJ), version d'étude du 27/05/2021.
- Raghavan M., Barocas S., Kleinberg J., Levy K. (2019) Mitigating bias in Algorithmic Hiring : Evaluating Claims and Practices, *Proceedings of the Conference on Fairness, Accountability, and Transparency*.
- Rich J. (2014). What Do Field Experiments of Discrimination in Markets Tell Us? A Meta Analysis of Studies Conducted since 2000, *IZA Discussion Paper*, No. 8584.
- M. Roberts, D. Driggs, M. Thorpe, J. Gilbey, M. Yeung, S. Ursprung, A. I. Aviles-Rivero, C. Etmann, C. McCague, L. Beer, J. R. Weir-McCall, Z. Teng, E. Gkrania-Klotsas, AIX-COVNET, J. H. F. Rudd, Evis Sala, C.-B. Schönlieb (2021), Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans, *Nature Machine Intelligence*, 3, pages 199–217.
- Verzelen N. (2012). Minimax risks for sparse regressions: Ultra-high dimensional phenomenons, *Electron. J. Statist.*, 6, 38 - 90.
- Zliobaité I. (2017). Measuring discrimination in algorithmic decision making, *Data Min. Knowl. Disc.*, 31, p 1060–89.

Bibliography

[1] Gelman A. and Hill J., *Data analysis using regression and multilevel/hierarchical models*, ch. 25, pp. 529–563, Cambridge University Press, 2007.

[2] H. Akaïke, *A new look at the statistical model identification*, IEEE Transactions on Automatic Control **19** (1974).

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J.L. Reyes-Ortiz, *Energy efficient smartphone-based activity recognition using fixed-point arithmetic*, Journal of Universal Computer Science. Special Issue in Ambient Assisted Living: Home Care **19** (2013).

[4] K. Bache and M. Lichman, *UCI machine learning repository*, 2013, http://archive.ics.uci.edu/ml.

[5] P. Besse, H. Milhem, O. Mestre, A. Dufour, and V. H. Peuch, *Comparaison de techniques de data mining pour l'adaptation statistique des prévisions d'ozone du modèle de chimie-transport mocage*, Pollution Atmosphérique **195** (2007), 285–292.

[6] G. Biau, A. Ficher, B. Guedj, and J. D. Malley, *Cobra: A nonlinear aggregation strategy*, Journal of Multivariate Analysis **146** (2016), 18–28.

[7] L. Breiman, *Bagging predictors*, Machine Learning **26** (1996), no. 2, 123–140.

[8] _____, *Random forests*, Machine Learning **45** (2001), 5–32.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and regression trees*, Wadsworth & Brooks, 1984.

[10] Rich. Caruana, N. Karampatziakis, and A. Yessenalina, *An empirical evaluation of supervised learning in high dimensions*, Proceedings of the 25th International Conference on Machine Learning (New York, NY, USA), ICML '08, ACM, 2008, pp. 96–103, ISBN 978-1-60558-205-4.

[11] Rubin D.B., *Multiple imputation for nonresponse in surveys*, Wiley, 1987.

[12] Stekhoven D.J. and Bühlmann P., *Missforest - nonparametric missing value imputation for mixed-type data*, Bioinformatics Advance Access (2011).

[13] David Donoho, *50 years of data science*, Princeton NJ, Tukey Centennial Workshop, 2015.

[14] B. Efron and R. Tibshirani, *Improvements on cross-validation: The .632+ bootstrap method*, Journal of the American Statistical Association **92** (1997), no. 438, 548–560.

[15] Hastie et al, *Imputing missing data for gene expression arrays*, Techn. rep., Division of Biostatistics, Stanford University, 1999.

[16] M Fernández-Delgado, E Cernadas, S Barro, and D Amorim, *Do we need hundreds of classifiers to solve real world classification problems?*, The journal of machine learning research **15** (2014), no. 1, 3133–3181.

[17] Y. Freund and R.E. Schapire, *Experiments with a new boosting algorithm*, Machine Learning: proceedings of the Thirteenth International Conference, Morgan Kaufman, 1996, San Francisco, pp. 148–156.

[18] J. H. Friedman, *Stochastic gradient boosting*, Computational Statisrics and Data Analysis **38** (2002), .

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016, http://www.deeplearningbook.org.

[20] David J. Hand, *Classifier technology and the illusion of progress*, Statist. Sci. **21** (2006), no. 1, 1–14.

[21] T. Hastie, R. Tibshirani, and J Friedman, *The elements of statistical learning : data mining, inference, and prediction*, Springer, 2009, Second edition.

[22] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Improving neural networks by preventing co-adaptation of feature detectors*, CoRR, abs/1207.0580 (2012).

[23] Honaker J., King G., and Blackwell M., *Amelia ii: A program for missing data*, Journal of statistical software **45** (2011), no. 7.

[24] D. Kingma and J. Ba, *Adam : a method for stochastic optimization*, Arxiv **1412.6980** (2014).

[25] Y. LeCun, L. Jackel, B. Boser, J. Denker, H. Graf, I. Guyon, D. Henderson, R. Howard, and W. Hubbard, *Handwritten digit recognition : Applications of neural networks chipsand automatic learning*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[26] M. Lichman, *UCI machine learning repository*, 2013, http://archive.ics.uci.edu/ml.

[27] Glasson Cicignani M. and Berchtold A., *Imputation de donnees manquantes : Comparaison de differentes approches*, 42e Journees de Statistique, 2010.

[28] C.L. Mallows, *Some comments on cp*, Technometrics **15** (1973), 661–675.

[29] Setiawan N.A., Venkatachalam P.A., and Hani A.F.M., *A comparative study of imputation methods to predict missing attribute values in coronary heart disease data set*, 4th Kuala Lumpur International Conference on Biomedical Engineering 2008 (University of Malaya Department of Biomedical Engineering Faculty of Engineering, ed.), vol. 21, Springer Berlin Heidelberg, 2008, pp. 266–269.

[30] Y. Nesterov, *A method of solving a complex programming problem with convergence rate o(1/k2)*, Soviet Mathematics Doklady **27** (1983), 372–376.

[31] B.T. Polyak, *Some methods of speeding up the convergence of iteration methods*, USSR Computational Mathematics and Mathematical Physics **4(5)** (1964), 1–17.

[32] Detrano R., Janosi A., Steinbrunn W., Pfisterer M., Schmid J., Sandhu S., Guppy K., Lee S., and Froelicher V., *International application of a new probability algorithm for the diagnosis of coronary artery disease*, American Journal of Cardiology **64** (1989), 304–310.

[33] Little R.J.A. and Rubin D.B., *Statistical analysis with missing data*, Wiley series in probability and statistics, 1987.

[34] B. Scholkopf and A. Smola, *Learning with kernels support vector machines, regularization, optimization and beyond*, MIT Press, 2002.

[35] G. Schwarz, *Estimating the dimension of a model*, Annals of Statistics **6** (1978), 461–464.

[36] E. Scornet, G. Biau, and J. P. Vert, *Consistency of random forests*, The Annals of Statistics **43** (2015), no. 4, 1716–1741.

[37] I. Sutskever, J. Martens, G.E. Dahl, and G.E. Hinton, *On the importance of initialization and momentum in deep learning*, ICML **28(3)** (2013), 1139–1147.

[38] R. Tibshirani, *Regression shrinkage and selection via the lasso*, J. Royal. Statist. Soc B **58** (1996), 267–288.

[39] M. J. van der Laan, E. C. Polley, and A. E. Hubbard, *Super learner*, Statistical Applications in Genetics and Molecular Biology **6:1** (2007).

[40] V.N. Vapnik, *Statistical learning theory*, Wiley Inter science, 1999.

[41] Grzymala Busse J. W., Grzymala Busse W. J., and Goodwin L. K., *Coping with missing attribute values based on closest fit in preterm birth data: A rough set approach*, Computational Intelligence **17** (2001), 425–434.

[42] Cleveland W.S. and Devlin S.J., *Locally-weighted regression: An approach to regression analysis by local fitting*, Journal of the American Statistical Association **83** (1988), no. 403, 596–610.